BERGISCHE
UNIVERSITÄT
WUPPERTAL

Generalized Algebraic Kernels and Multipole Expansions for massively parallel Vortex Particle Methods

Zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften
am Fachbereich Mathematik der
Bergischen Universität Wuppertal
genehmigte

Dissertation

von

Dipl.-Math. Robert Speck

Tag der mündlichen Prüfung: 1. Juli 2011

Gutachter: Prof. Dr. Rolf Krause
Prof. Dr. Dr. Thomas Lippert
Prof. Lorena Barba, PhD

TO THOSE WE HAVE LOST
AND TO THOSE LEFT BEHIND

Abstract – Regularized vortex particle methods offer an appealing alternative to common mesh-based numerical methods for simulating vortex-driven fluid flows. While inherently mesh-free and adaptive, a stable implementation using particles for discretizing the vorticity field must provide a scheme for treating the overlap condition, which is required for convergent regularized vortex particle methods. Moreover, the use of particles leads to an N-body problem. By the means of fast, multipole-based summation techniques, the unfavorable yet intrinsic $\mathcal{O}(N^2)$-complexity of these problems can be reduced to at least $\mathcal{O}(N \log N)$. However, this approach requires a thorough and challenging analysis of the underlying regularized smoothing kernels. We introduce a novel class of algebraic kernels, analyze its properties and formulate a decomposition theorem, which radically simplifies the theory of multipole expansions for this case. This decomposition is of great help for the convergence analysis of the multipole series and an in-depth error estimation of the remainder. We use these results to implement a massively parallel Barnes-Hut tree code with $\mathcal{O}(N \log N)$-complexity, which can perform complex simulations with up to 10^8 particles routinely. A thorough investigation shows excellent scalability up to 8192 cores on the IBM Blue Gene/P system JUGENE at Jülich Supercomputing Centre. We demonstrate the code's capabilities along different numerical examples, including the dynamics of two merging vortex rings. In addition, we extend the tree code to account for the overlap condition using the concept of remeshing, thus providing a promising and mathematically well-grounded alternative to standard mesh-based algorithms.

Zusammenfassung – Regularisierte Vortex-Partikel-Methoden bilden eine interessante Alternative zu gitterbasierten numerischen Methoden zur Simulation von vortex-dominierten Flüssigkeitsströmungen. Eine stabile Implementierung dieses intrinsisch gitterfreien und adaptiven Ansatzes durch die Diskretisierung des Vortex-Feldes benötigt eine Schema zur Behandlung der Überlapp-Bedingung, welche für eine konvergente regularisierte Vortex-Partikel-Methode erforderlich ist. Desweiteren führt der Gebrauch von Partikeln zu einem N-Körper-Problem. Schnelle, multipol-basierte Summationstechniken können die ungünstige aber intrinsisch verankerte $\mathcal{O}(N^2)$-Komplexität dieser Probleme auf mindestens $\mathcal{O}(N \log N)$ reduzieren. Dieser Ansatz benötigt jedoch eine genaue und herausfordernde Analyse der zugrunde liegenden Regularisierungskerne. Wir führen eine neue Klasse algebraischer Kerne ein, analysieren ihre Eigenschaften und leiten einen Zerlegungssatz her, welcher die Theorie der Multipol-Entwicklungen für diesen Fall radikal vereinfacht. Diese Zerlegung ist von großem Nutzen bei der Konvergenzanalyse der Multipolreihe und einer detaillierten Fehlerabschätzung des Restgliedes. Wir nutzen diese Ergebnisse zur Implementation eines massivparallelen Barnes-Hut Tree Codes mit $\mathcal{O}(N \log N)$-Komplexität, welcher komplexe Simulationen mit bis zu 10^8 Partikeln problemlos durchführen kann. Eine genaue Analyse zeigt exzellente Skalierbarkeit auf bis zu 8192 Kernen des IBM Blue Gene/P Systems JUGENE am Jülich Supercomputing Centre. Wir demonstrieren die Fähigkeiten des Codes anhand verschiedener numerischer Beispiele, unter anderem anhand der Dynamik zweier fusionierender Vortex-Ringe. Zusätzlich erweitern wir den Tree Code derart, dass die Überlapp-Bedingung mit Hilfe des Remeshing-Konzepts eingehalten werden kann, so dass der Code eine vielversprechende und mathematisch fundierte Alternative zu gitterbasierten Standard-Algorithmen darstellt.

Contents

List of Figures

1 Introduction

"Each rotating element of fluid (a) implies in each other element (b) of the same fluid mass a velocity whose direction is perpendicular to the plane through (b) and the axis of rotation of (a)."

Hermann Ludwig Ferdinand von Helmholtz, 1867 [1]

1.1 The Navier-Stokes equations

Fluid dynamics are an important topic in today's applied sciences. The modeling of fluid flows and their properties is an active field of research in engineering, computer science and mathematics. From the design of airplanes to the development of massively parallel algorithms and the analysis of partial differential equations, the broad field of fluid dynamics has great influence on many different scientific disciplines.

The mathematical basis for modeling and analyzing fluid flows are the Navier-Stokes equations. Based on Newton's second law of motion, the Navier-Stokes equations express the conservation of momentum and mass [2, 3, 4, 5]. Although composed of molecules and atoms, this analytical approach interprets the fluid as moving, continuous media. This principle is known as the continuum hypothesis. In the formulation we will use at the beginning of this work, the velocity field and pressure term describe the motion of incompressible viscous fluid flows in three spatial dimensions. While inviscid fluid flows are often called 'ideal flows', viscosity provides a measure for fluid friction and its properties under shear or tensile stress. An incompressible flow has the property that the density of a fluid element moving along its trajectory does not change in time. To some extent, this assumption is a simplification of physical reality, but it is of great help for the treatment of equations governing fluid flows. Mathematically, this is expressed by a solenoidal velocity field, i.e. one that is divergence-free.

An extensive and in-depth overview of fundamental laws and properties of fluids can be found in the classical work 'Hydrodynamics' by Sir Horace Lamb [6]. In the following definition we cite the Navier-Stokes equations along with some terminological prepositions.

Definition 1.1.1 (Navier-Stokes equations)

a) For a time $T \in (0, \infty)$ and $d \in \mathbb{N}_0$ we identify the *velocity field* in d spatial dimensions with $u(x,t) : \mathbb{R}^d \times [0,T] \to \mathbb{R}^d$. Furthermore, we describe the *pressure field* using $p(x,t) : \mathbb{R}^d \times [0,T] \to \mathbb{R}$.

b) The *Navier-Stokes equations* for incompressible viscous unsteady fluid flow in a three-dimensional inertial frame of reference are given by

$$\frac{\partial u}{\partial t} + u \cdot \nabla u = -\nabla p + \nu \, \Delta u,$$
$$\nabla \cdot u = 0.$$

The first equation represents the balance or conservation of momentum, while the second represents conservation of mass. A divergence-free vector field is often called *solenoidal*. The constant ν represents the *kinematic viscosity*.

In three spatial dimensions it is still an open problem to prove or disprove that analytical solutions for these equations exist for all times. While in two dimensions this question of existence for the above problem has been solved satisfactorily, in three dimensions the problem is proven to be well posed – i.e. existence of a unique solution depending continuously on the initial data – only for short time scales [7, 8]. For an analysis of the problem, suitable boundary conditions have to be supplemented. For instance, a scenario with inviscid flows, i.e. $\nu = 0$, may require that the fluid moves tangentially to the boundary, but is not allowed to cross it. For $\nu > 0$, this is not sufficient as the number of derivatives of u is increased from one to two. Therefore, an appropriate boundary condition for u is the so-called 'no-slip'-condition, i.e. $u \equiv 0$ on solid walls at rest [3].

Since analytical solutions of the Navier-Stokes equations are only known in very rare cases (for details see [9, 10] and references therein) the theoretical investigation of fluid flows is based on numerical methods. Experimental setups are often expensive and hard to construct, so that numerical simulations are the only way to shed light on many different phenomena in fluid dynamics. They can be subjected to in-depth analysis and their results can be reproduced easily. In contrast to experimental setups, the initial access to numerical methods is much more easier, relying on well-grounded mathematical and algorithmic theories. The basis for each numerical scheme is the initial question of how to discretize the continuous media. This question is strongly coupled to the underlying variables of the equations and it may turn out to be advantageous to rethink often used formulations, as we will see in the next section.

The Navier-Stokes equations in the above given form consist of four equations for two variables u and p, where u is composed of three scalar variables u_1, u_2, u_3. Therefore, this set of equations is sufficient to describe a fluid motion depending on the velocity u and the pressure p completely. The variables u and p are often called 'primitive variables' [11, 12]. An alternative formulation uses the so-called 'vorticity field' $\omega = \nabla \times u$, being a 'non-primitive variable'. Applying the curl operator to the equation of motions of Definition 1.1.1 yields

the so-called 'vorticity-velocity equation'. Here, the pressure term is decoupled from the velocity field and can be recovered using a Possion equation [13], but it does not have to be derived explicitly. We trace the transition from Navier-Stokes equations to the vorticity-velocity equation in the next section in detail and highlight the advantages of this alternative formulation.

1.2 The vorticity-velocity equation

Vortex-driven flows play an important role in many different aspects of fluid dynamics and applications. Especially for an industrial development and design of windmills or airplanes, the influence of vortical flows have to be considered carefully. In particular, the vorticity field and its dynamics are the basis for turbulent fluid flows. From a mathematical point of view, the vorticity field is given by the rotation of the velocity field.

Definition 1.2.1 (Vorticity)
In three spatial dimensions the vorticity field ω is defined as the curl of the velocity field, i.e. $\omega(x,t) := \nabla \times u(x,t)$ for $x \in \mathbb{R}^3, t \in [0,T]$. This representation is valid in two spatial dimensions as well if we assume $u = (u_1, u_2, 0)^T$ and $\omega = (0, 0, \omega_3)^T$.

Comprehensive introductions in the context of vortex-driven flows can be found in [14, 15, 16]. Now, for the deduction of the vorticity-velocity equation we make use of vector calculus identities and notations described in Appendix A. Most of the upcoming manipulations are taken from [2, 3].

Before applying the curl operator to the Navier-Stokes equations, we use the Identities (A.4) and (A.5) to rewrite the momentum equation of Definition 1.1.1 as

$$\frac{\partial u}{\partial t} + \omega \times u = -\nabla \left(p + \frac{u \cdot u}{2} \right) + \nu \, \Delta u$$

using the vorticity field $\omega = \nabla \times u$. We now take the curl of this equation and obtain with Equation (A.2)

$$\frac{\partial \omega}{\partial t} + \nabla \times (\omega \times u) = \nu \left(\nabla \times (\Delta u) \right).$$

Here we can see that the pressure term has already vanished. We rewrite the term on the right-hand side by using Identity (A.8), which yields

$$\nabla \times (\Delta u) = \nabla \times (\nabla(\nabla \cdot u)) - \nabla \times (\nabla \times \omega).$$

Since the conservation of mass of the Navier-Stokes equations requires a divergence-free velocity field u this reduces to

$$\nabla \times (\Delta u) = -\nabla \times (\nabla \times \omega).$$

With $\nabla \cdot \omega = 0$ by Equation (A.1), we obtain

$$\nabla \times (\Delta u) = \Delta \omega.$$

Therefore, the curl of Laplacian of a divergence-free velocity field equals the Laplacian of the vorticity.

For the left-hand side we make use of Equation (A.6), which yields

$$\nabla \times (\omega \times u) = \omega(\nabla \cdot u) - u(\nabla \cdot \omega) + (u \cdot \nabla)\omega - (\omega \cdot \nabla)u.$$

Again, since $\nabla \cdot \omega = 0$ by Equation (A.1) and $\nabla \cdot u = 0$ by Definition 1.1.1 this reduces to

$$\nabla \times (\omega \times u) = (u \cdot \nabla)\omega - (\omega \cdot \nabla)u,$$

so that we finally obtain

$$\frac{\partial \omega}{\partial t} + (u \cdot \nabla)\omega = (\omega \cdot \nabla)u + \nu\, \Delta \omega.$$

This equation is the so-called 'vorticity-velocity equation'.

Definition 1.2.2 (Vorticity-velocity equation)
For a solenoidal velocity field u with $\nabla \cdot u = 0$ and vorticity field $\omega = \nabla \times u$ the equation

$$\frac{\partial \omega}{\partial t} + (u \cdot \nabla)\omega = (\omega \cdot \nabla)u + \nu \Delta \omega$$

is called *vorticity-velocity equation*. The first term on the right-hand side is called *stretching term*, the second one *diffusion term*.

Remark 1.2.3
We note that in two spatial dimensions and for $\nu = 0$ this formulation reduces to the standard 2D Euler equation

$$\frac{\partial \omega}{\partial t} + (u \cdot \nabla)\omega = 0,$$

since the stretching term vanishes if we reduce the three-dimensional problem to a two-dimensional one, taking $u = (u_1, u_2, 0)$ and $\omega = (0, 0, \omega_3)$. The stretching term, manipulating the vorticity field in the direction of the stretching, is the driving force for turbulent fluids. Therefore, the vorticity-velocity equation shows that turbulence can only occur in three-dimensional flows.

The vorticity-velocity equation in the present form is the so-called 'classical' equation [17]. With Equation (A.7) it is possible to rewrite the vorticity-velocity equation and obtain two alternative formulations.

Theorem 1.2.4
For a solenoidal velocity field u with $\nabla \cdot u = 0$ and vorticity field $\omega = \nabla \times u$ the vorticity-velocity equation reads

$$\begin{aligned}\frac{\partial \omega}{\partial t} + (u \cdot \nabla)\omega &= (\omega \cdot \nabla^T)u + \nu\Delta\omega \\ &= \frac{1}{2}(\omega \cdot (\nabla + \nabla^T))u + \nu\Delta\omega.\end{aligned}$$

Proof. This result follows directly from Equations (A.3) and (A.7). □

The vorticity-velocity equation does not contain an explicit statement for the divergence of u like the Navier-Stokes equations of Definition 1.1.1. However, as noted in Definition 1.2.2 and Theorem 1.2.4, a divergence-free velocity field is required for most of the manipulations that lead to the vorticity-velocity equation.

The equation of Definition 1.2.2 is a main ingredient of vortex methods. From a practical perspective, the reformulation of the Navier-Stokes equations in terms of vorticity field ω and velocity field u has one important advantage: using a given vorticity field and a solenoidal stream function as basis for the velocity, we can state an explicit and analytic form for u. To this end, we decompose the velocity field into a solenoidal (and rotational) component u_ω and a potential (and irrotational) component u_p. We obtain

$$u = u_\omega + u_p,$$

with

$$\nabla \cdot u_\omega = \nabla \times u_p = 0.$$

The first component accounts for the vortical part induced by the vorticity field in unbounded space with $\omega = \nabla \times u_\omega$, and the second one can be described in terms of a potential Φ with $u_p = \nabla\Phi$, accounting for adequate boundary conditions and potential flow components [13, 18]. This decomposition is called Helmholtz decomposition [3, 13].

Furthermore, assuming the existence of a divergence-free vector stream function ψ with

$$u_\omega = \nabla \times \psi$$

we obtain

$$\Delta\psi = -\omega.$$

Recalling that the Green's function for $-\Delta$ in three-dimensional unbounded domains is defined as $G(x) := \frac{1}{4\pi|x|}$ we get [17, 19]

$$\psi(x,t) = G(x) * \omega(x,t) = \int G(x-y)\omega(y,t)\mathrm{d}y$$

with the common convolution product '$*$'. We apply this expression to u and obtain the following theorem. Its content bears strong similarities to the so-called 'Biot-Savart law' – a well-known result in the field of electromagnetism [17, 20], coupling the analytic expression of a magnetic field to an electric current. This interesting connection between electromagnetism and vortex dynamics motivates the name of the kernel K used in the following theorem.

Theorem 1.2.5 (Biot-Savart kernel)
The solenoidal field u_ω in three spatial dimensions can be written as

$$u_\omega(x,t) = \nabla \times \psi(x,t) = \int K(x-y) \times \omega(y,t) \mathrm{d}y,$$

where the function

$$K(x) := -\frac{x}{4\pi|x|^3}$$

is called 'Biot-Savart kernel'. We write $u_\omega := K \circledast \omega$.

Proof. A discussion of this relation can be found in [16, 21, 22]. □

While u_ω is induced by the vorticity field in an unbounded space, the potential component u_p fulfills the current boundary conditions but vanishes identically for an unbounded domain with no interior boundaries. The stream function ψ is solenoidal if the velocity is assumed to decay to zero as the boundaries of the domain extend to infinity [2, 13]. Throughout this work we assume $u \equiv u_\omega$, using unbounded domains with no interior boundaries only. Equipped with these boundary assumptions, the vorticity-velocity equation of Definition 1.2.2 and the Biot-Savart formulation of Theorem 1.2.5 are the main principles of vortex methods.

The transition from the classical Navier-Stokes equations to the alternative vorticity-velocity formulation increases the number of unknowns by two, since the one-dimensional pressure term is replaced by the three-dimensional vorticity field. However, the separation of the pressure term and the connection of velocity and vorticity on an analytical level have appealing advantages. As noted in [23], vortex dynamics are already well understood for incompressible viscous flows. Furthermore, for vortex-dominated flows the formulation via the connection of vorticity and velocity is a more natural representation of physical reality. In the case of numerical simulations, boundary conditions at infinity are easier to realize for the vorticity than for the pressure [23]. As noted in [11], the elimination of the pressure term also leads to an increased numerical stability, removing the artificial sensitivity of numerical systems that is induced by discretization operators.

Most numerical implementations using the classical Navier-Stokes equations rely on a mesh-based discretization of the region of interest [24, 25]. Intriguingly, the explicit dependence

of u on the vorticity field as stated in Theorem 1.2.5 yields an promising alternative. Using a quadrature approach, the integral in Theorem 1.2.5 can be discretized with quadrature points. These points can be seen as particles discretizing the vorticity field. This idea yields a fascinating field of research: the vortex particle methods. We emphasize that these particles should not be equalized with actual physical particles like molecules or atoms, but rather with discretization nodes. However, we will see that the introduction of particles induces many problems that heavily influence the development of a numerical method for simulating vortex-driven flows. Besides theoretical and design issues, a major challenge is the algorithmic complexity of the problem. While the introducton of regularization and location processing techniques ensures analytical convergence, the mutual interaction of all particles impedes its practical use. The emergence of this principle, the so-called 'N-body problem', closes the circle to the introductory citation of Helmholtz: each particle has an influence on all other particles of the same fluid. Therefore, the usage of particles creates a strong need for fast summation algorithms, in order to make simulations with millions of particles economically feasible.

It is the goal of this work to formulate a fast, multipole-based summation technique for vortex particle methods. In order to perform 'fast summations' in the context of vortex particle methods, we will need to analyze the exact kernels that represent the actual particle discretization. This will be done by the introduction of a novel class of regularized algebraic smoothing kernels, which will turn out to be very helpful for analyzing multipole expansions for vortex particle methods. The drawback of particle regularization is the emergence of an additional convergence requirement: the overlap condition. We will review a method to satisfy this requirement and show how this technique can be implemented efficiently, even in parallel. With stable and fast simulation algorithms at hand, the number of particles can be increased considerably, exceeding the available storage on normal computers soon. This motivates the usage of massively parallel supercomputers, posing additional challenges on the implementation of these summation schemes.

The structure of this thesis is as follows: Chapters 2 to 4 start with a design decision based on the result of the previous chapter. At the beginning of each of these chapters, we will briefly weigh pros and cons of two different approaches and motivate our decision. The consequences of this decision will determine the content of the respective chapter. At the end of each of the three chapters, we will restate the decision, summarize the results and prepare the choice of the following chapter.

The content of this thesis is organized as follows: In Chapter 2 we will provide a detailed description of regularized vortex particle methods and their convergence statement. A commonly used location processing technique ensuring stability and its extension covering viscous effects will be reviewed. The topics of Chapter 3 are the definition and extensive analysis of a new class of algebraic smoothing kernels. A decomposition theorem will facilitate access to the theory of multipole expansions. It will allow us to show convergence of the expansion in a very convenient way and to derive a new explicit upper bound for the expansion re-

mainder. In Chapter 4, we will apply the theoretical approach of Chapter 3 and describe a parallel implementation of a multipole-based Barnes-Hut tree code. By means of extensive benchmarks, we will demonstrate its capabilities on the IBM Blue Gene/P system JUGENE at Jülich Supercomputing Centre (JSC) and show applications using a spherical vortex sheet and two merging vortex rings. Finally, we will summarize our results and give an outlook on further improvements in Chapter 5.

2 Fundamentals of regularized vortex particle methods

It is one of the great advantages of the vorticity-velocity formulation that Theorem 1.2.5 provides a straightforward way to analytically derive the velocity field u out of a known vorticity distribution. However, for numerical applications a discretization scheme is necessary. We have already mentioned the introduction of particles in the last chapter, but following the classical Navier-Stokes equations, mesh-based methods are available here as well. Figure 2.1 illustrates these two fundamentally different approaches. We highlight and briefly weigh pros and cons of both concepts in the following.

Mesh-based methods – Finite differences, elements or volumes are common mesh-based methods, especially used in classical computational fluid dynamics [24, 25]. These methods rely on an underlying mesh structure for the discretization of the vorticity-velocity equation and the Poisson equation for the stream function [26, 27, 28]. The differential operators are then translated onto the mesh, where their evaluation can be performed quickly and accurately. To obtain a certian degree of adaptivity, advanced and complex methods such as adaptive mesh refinement are commonly applied [29].

Particle-based methods – For this approach, the flow field is discretized using particles that occupy a particular area in the region of interest, transporting quantities like volume, velocity or vorticity. In contrast to mesh-based methods, this interpretation links particle-based approaches directly to the underlying physics. While other methods in this context like

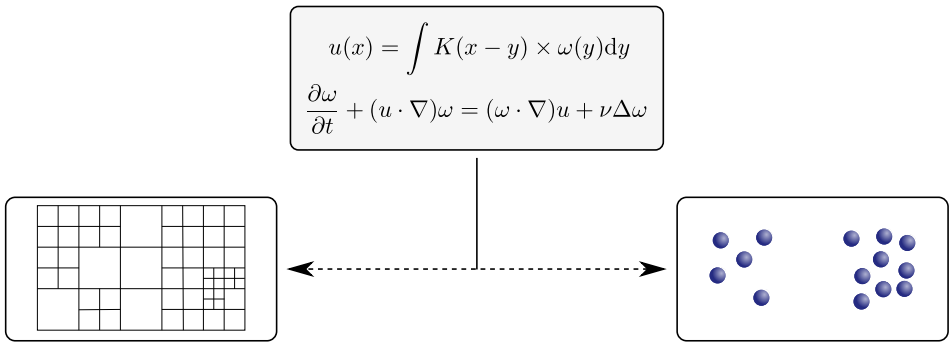


Figure 2.1: Discretization using mesh-based or particle-based methods?

smoothed particle hydrodynamics [30, 31] focus on the original Navier-Stokes equations and discretize the differential operators directly, the vortex particle methods rely on a discretization of the vorticity field.

Despite undeniable advantages like a large range of fast algorithms and well-grounded mathematical backgrounds, mesh-based approaches often suffer from an very inflexible handling of the mesh structure. For resolving complex flow patterns, adaptive mesh refinement is mandatory. In turn, this implies not only the handling of often complicated data structures but also has strong influence on the properties and computational efficiency of the underlying methods and implementations. In particular, this is crucial in case of turbulent flows or boundaries at infinity. On the other hand, particle-based methods share an intrinsic adaptivity, since computational elements exists only where the corresponding fields are non-zero. Furthermore, boundaries at infinity are automatically satisfied.

In this chapter, we review the concept of regularized vortex particles and discuss the implications arising from this approach. The goal is the formulation of a numerical particle-based vortex method, reliably simulating dynamical vorticity-driven flows.

To this end, we will discuss in the first section how the discretization by regularized particles leads to a numerically stable vortex particle method and to a consistent convergence result for the inviscid case. However, it will turn out that further effort is required to control the particle alignment and emerging field distortions. In the second section, we will review one commonly used location processing technique, which is able to efficiently restore numerical stability, thus ensuring convergence for longer simulation times. Furthermore, this technique can be extended to account for viscous effects, turning the additional expenses into a tangible advantage.

2.1 Discretization using regularized particles

A very detailed and accessible overview of particle discretization in the context of vortex methods can be found in works by Winckelmans and Leonard [17, 32]. We review their concept in this section. The basic idea of vortex particle methods is the discretization of the support of the vorticity field by means of cells with size $h > 0$, in which the circulation is concentrated on a single particle. Let $\mathcal{P} = \{1, ..., N\} \subset \mathbb{N}$ be a set of particle indices, each of them provided with a position $x_p \in \mathbb{R}^3$ and a time-independent volume $\mathrm{vol}_p \in (0, \infty)$, $p \in \mathcal{P}$. For a classical decomposition of a three-dimensional region of interest into cubic and disjoint cells with width h, we further set $\mathrm{vol}_p \equiv h^3$ for all $p \in \mathcal{P}$.

2.1.1 The point vortex method

As a first and very simple approach, we choose a discretization $\tilde{\omega}$ of the vorticity field ω as

$$\begin{aligned}\tilde{\omega}(x,t) &= \sum_{p\in\mathcal{P}} \delta(x - x_p(t)) \cdot \omega(x_p(t),t)\mathrm{vol}_p \\ &= \sum_{p\in\mathcal{P}} \delta(x - x_p(t)) \cdot \alpha_p(t).\end{aligned}$$

Here, $\delta(x)$ denotes the 3D δ-function, $x_p(t)$ is the approximated trajectory resulting from a discretized version $\tilde{u}$ of u at time $t \geq 0$ and

$$\alpha_p(t) := \omega(x_p(t),t)\mathrm{vol}_p$$

is the so-called 'strength vector' of a particle $p \in \mathcal{P}$. A detailed mathematical justification for this approximation using singular particles can be found in [13, 33, 34] and the references therein.

Applying this discretization to Theorem 1.2.5, we can now obtain $\tilde{u}$ as an approximation of the velocity field u, so that

$$\begin{aligned}\tilde{u}(x,t) =&\ \nabla \times \tilde{\psi}(x,t) = \nabla \times (G(x) * \tilde{\omega}(x,t)) = \nabla G(x) \circledast \tilde{\omega}(x,t) \\ =&\ \sum_{p\in\mathcal{P}} \nabla G(x - x_p(t)) \times \alpha_p(t) = \sum_{p\in\mathcal{P}} K(x - x_p(t)) \times \alpha_p(t).\end{aligned}$$

This is the discretized version of Theorem 1.2.5 and it is the basis for the so-called 'point vortex method' [13, 35]. However, examining

$$K(x - x_p(t)) = -\frac{1}{4\pi}\frac{x - x_p(t)}{|x - x_p(t)|^3},$$

it is obvious that the singularity of the function K may lead to numerical instabilities when particles move towards each other. First used in 1931 by Rosenhead [36] for vortex sheet dynamics in two spatial dimensions, this approach has been considered as non-convergent for a long time [37]. As an alternative, the 'vortex blob method' or 'regularized vortex method' [21] introduced particles with a finite core size σ, circumventing the singularity of the kernel K. Ironically, this approach and its convergence framework led in turn to a convergence result for the point vortex method. In 1990, Hou, Goodmann and Lowengrub [37, 38] were able to prove convergence for the 3D vortex method without smoothing under the assumption that the vorticity is smooth and has compact support. However, for particles with distance h this result implies that the point vortex method is only of order $\mathcal{O}(h^2)$ (see e.g. [13] for an overview), while a careful choice of the regularization kernel implies vortex methods of arbitrary order.

2.1.2 Smoothing kernels

The basis for the regularized vortex particle method is the choice of the regularized smoothing kernel, which transforms point particles into 'blobs' with finite core size. The following definition characterizes these kernels.

Definition 2.1.1 (Regularized smoothing kernels)

a) A function $\zeta : \Omega \to \mathbb{R}$ for $\Omega = \mathbb{R}^d$, $d = 2, 3$ with $\zeta \in C^\infty(\Omega)$ is called *smoothing kernel* if it satisfies the *normalization constraint*

$$\int_\Omega \zeta(y)\mathrm{d}y = 1.$$

b) A function $\zeta_\sigma : \Omega \to \mathbb{R}$ for $\Omega = \mathbb{R}^d$, $d = 2, 3$, with

$$\zeta_\sigma(x) = \frac{1}{\sigma^3}\,\zeta\left(\frac{x}{\sigma}\right),$$

is called *regularized smoothing kernel* with core size $\sigma > 0$.

c) A smoothing kernel $\zeta = \zeta^{(r)}$ is of order $r \in \mathbb{N}_0$ if the following *moment conditions* hold:

$$\int_\Omega \zeta(y)\, y^\beta \mathrm{d}y = 0, \text{ for all } 0 < |\beta| \leq r-1 \text{ with multi-index } \beta,$$
$$\int_\Omega |\zeta(y)|\, |y|^r \mathrm{d}y < \infty.$$

For details on multi-index notation we refer to Definition A.3.

Remark 2.1.2

We note that normalization constraint and moment conditions are written in Cartesian coordinates as stated in [13, 39]. For three-dimensional radially symmetric smoothing kernels, i.e. for $\Omega = \mathbb{R}^3$ and

$$\zeta(x) = \zeta(|x|) =: \zeta(\rho),$$

these conditions read [17]

$$\int_0^\infty \zeta(\rho)\,\rho^2 \mathrm{d}\rho = \frac{1}{4\pi},$$
$$\int_0^\infty \zeta(\rho)\,\rho^{2+s} \mathrm{d}\rho = 0 \quad \text{for all } 2 \leq s \leq r-1,\ s \text{ even},$$
$$\int_0^\infty |\zeta(\rho)|\,\rho^{2+r} \mathrm{d}\rho < \infty.$$

We refer to these integral expressions as moment integrals.

Before we analyze these definitions, we apply the regularized smoothing kernel to singular particles, thus replacing the singular function K by a regularized function K_σ. We convolve the δ-function in $\tilde{\omega}$ with ζ_σ to obtain the method of regularized vortex particles.

Theorem 2.1.3
For a discretized vorticity field $\tilde{\omega}$ and a regularized smoothing kernel ζ_σ, we obtain

$$\begin{aligned}
\tilde{\omega}_\sigma(x,t) &= \sum_{p\in\mathcal{P}} \zeta_\sigma(x - x_p(t)) \cdot \alpha_p(t), \\
\tilde{\psi}_\sigma(x,t) &= \sum_{p\in\mathcal{P}} G_\sigma(x - x_p(t)) \cdot \alpha_p(t), \\
\tilde{u}_\sigma(x,t) &= \sum_{p\in\mathcal{P}} K_\sigma(x - x_p(t)) \times \alpha_p(t),
\end{aligned}$$

where $G_\sigma(x) = 1/\sigma \cdot G(|x|/\sigma)$ and $K_\sigma = K * \zeta_\sigma$.

Proof. The deduction of these equations is straightforward [17]. The convolution of the δ-function with ζ_σ yields

$$\tilde{\omega}_\sigma(x,t) = \tilde{\omega}(x,t) * \zeta_\sigma(x) = \sum_{p\in\mathcal{P}} \zeta_\sigma(x - x_p(t)) \cdot \alpha_p(t)$$

and therefore $\Delta\tilde{\psi}_\sigma = -\tilde{\omega}_\sigma$. Defining $G(\rho)$ such that

$$-\zeta(\rho) = \Delta G(\rho) = \frac{1}{\rho}\frac{\mathrm{d}^2}{\mathrm{d}\rho^2}\left(\rho\, G(\rho)\right),$$

we obtain

$$\tilde{\psi}_\sigma(x,t) = G(x) * \tilde{\omega}_\sigma(x,t) = G_\sigma(x) * \tilde{\omega}(x,t) = \sum_{p\in\mathcal{P}} G_\sigma(x - x_p(t)) \cdot \alpha_p(t)$$

with $G_\sigma(x) = 1/\sigma \cdot G(|x|/\sigma)$. This leads to

$$\begin{aligned}
\tilde{u}_\sigma(x,t) = \nabla \times \tilde{\psi}_\sigma(x,t) &= \sum_{p\in\mathcal{P}} \nabla G_\sigma(x - x_p(t)) \times \alpha_p(t) \\
&= \sum_{p\in\mathcal{P}} K_\sigma(x - x_p(t)) \times \alpha_p(t),
\end{aligned}$$

and also $\tilde{u}_\sigma(x,t) = K(x) \circledast \tilde{\omega}_\sigma(x,t) = K_\sigma(x) \circledast \tilde{\omega}(x,t)$. □

Remark 2.1.4
For an actual calculation of K_σ, we note that for $K = \nabla G$ and a smoothing kernel ζ we have [13, 17]

$$K_\sigma(x) = \frac{x}{|x|^3} \cdot f\left(\frac{|x|}{\sigma}\right)$$

with

$$f(\rho) = \int_0^\rho \zeta(s)\, s^2 \,\mathrm{d}s.$$

Both formulas can be used to derive K directly out of a given smoothing kernel ζ.

Figuratively speaking, Theorem 2.1.3 can be seen as the discretized and regularized version of Theorem 1.2.5. As the definition of $\tilde{\omega}_\sigma$ shows, singular vortices are now replaced by smeared, regularized vortex blobs with core size σ. The normalization constraint ensures the conservation of total vorticity and can therefore be seen as a moment condition of order zero. The other moment conditions are relevant for the conservation of higher momenta like linear and angular momentum [13, 21, 40, 41, 42]. The moment conditions are directly coupled to the convergence of the method, as we will see in Section 2.1.4: higher order smoothing functions yield a faster convergence rate, depending on the choice of σ. However, the conservation of higher momenta comes with a loss of positivity, inducing opposite signs of the smoothed vorticity field. For positive smoothing kernels ζ we find $r \leq 2$.

Many different smoothing kernels are available, each defining an individual three-dimensional regularized vortex particle method. We cite the two most often used approaches here explicitly.

a) Gaussian kernels [17, 43, 44] of order $r = 2$:

$$\zeta^{(2)}(\rho) = \frac{3}{4\pi} \exp\left(-\rho^3\right) \text{ and } \zeta^{(2)}(\rho) = \left(\frac{2}{\pi}\right)^{\frac{1}{2}} \exp\left(-\frac{\rho^2}{2}\right).$$

b) Algebraic kernels [32, 45] of order $r = 0$ and $r = 2$:

$$\zeta^{(0)}(\rho) = \frac{3}{4\pi} \frac{1}{(\rho^2+1)^{\frac{5}{2}}} \text{ and } \zeta^{(2)}(\rho) = \frac{15}{8\pi} \frac{1}{(\rho^2+1)^{\frac{7}{2}}}.$$

Additional smoothing kernels can be found in [43, 46, 47]. Moreover, these references include schemes to create higher order (even infinite order) kernels. For a numerical computation, it is crucial that the evaluation of these kernels can be performed as fast as possible. Here, algebraic kernels are much more convenient to use, especially in comparison to Gaussian smoothing [17]. In Section 3.1, we will introduce a new method for creating algebraic kernels of arbitrary order, extending the approach by Rosenhead [45] and Winckelmans [32]. This approach will turn out to be very suitable for rapid numerical calculations.

2.1.3 The equations of motion

For a preliminary numerical implementation of a regularized vortex particle method, evolution equations for a time integration scheme are required. For each particle $p \in \mathcal{P}$ two equations are necessary. The first one updates the particle position x_p and the second one satisfies the vorticity-velocity equation of Definition 1.2.2 to update the strength vector α_p.

Definition 2.1.5 (Equations of motion)
The following equations are the *equations of motion* for regularized vortex particle methods:

a) the *convection equation*

$$\frac{\mathrm{d}x_p(t)}{\mathrm{d}t} = \tilde{u}_\sigma(x_p(t), t),$$

b) and the *stretching and diffusion equation*

$$\frac{\mathrm{d}\alpha_p(t)}{\mathrm{d}t} = (\alpha_p(t) \cdot \nabla)\, \tilde{u}_\sigma(x_p(t), t) + \nu\, \Delta\alpha_p(x_p(t), t).$$

The second equation for α_p can be rewritten in two different ways [17, 35]. Initially, Definition 1.2.2 suggests the so-called 'classical scheme'

$$\frac{\mathrm{d}\alpha_p(t)}{\mathrm{d}t} = (\alpha_p(t) \cdot \nabla)\, \tilde{u}_\sigma(x_p(t), t) + \nu\, \Delta\alpha_p(x_p(t), t),$$

as stated in the preceding definition. Theorem 1.2.4 leads to two alternatives: first, the so-called 'transpose scheme' with

$$\frac{\mathrm{d}\alpha_p(t)}{\mathrm{d}t} = (\alpha_p(t) \cdot \nabla^T)\, \tilde{u}_\sigma(x_p(t), t) + \nu\, \Delta\alpha_p(x_p(t), t)$$

and second, the 'mixed scheme' with

$$\frac{\mathrm{d}\alpha_p(t)}{\mathrm{d}t} = \frac{1}{2}(\alpha_p(t) \cdot (\nabla + \nabla^T))\, \tilde{u}_\sigma(x_p(t), t) + \nu\, \Delta\alpha_p(x_p(t), t).$$

These schemes were identical for a three-dimensional vortex particle method if we could guarantee $\tilde{\omega}_\sigma = \nabla \times \tilde{u}_\sigma$. However, as noted in [17], we have

$$\nabla \times \tilde{u}_\sigma = \nabla \times (\nabla \times \tilde{\psi}_\sigma) = -\Delta\tilde{\psi}_\sigma + \nabla(\nabla \cdot \tilde{\psi}_\sigma) = \tilde{\omega}_\sigma + \nabla(\nabla \cdot \tilde{\psi}_\sigma)$$

and since $\tilde{\psi}_\sigma$ is not divergence-free (in contrast to its continuous, non-regularized counterpart ψ), we have to notice $\tilde{\omega}_\sigma \neq \nabla \times \tilde{u}_\sigma$. Even worse, the discretized vorticity field $\tilde{\omega}_\sigma$ itself is not divergence-free, so that it may not be a good representation of ω after some time steps.

The classical scheme – although not conservative in terms of total vorticity [17] – performs rather well regarding this problem. Initial disturbances of the divergence-free constraint are not further amplified, i.e. for the divergence of the discretized vorticity field $\tilde{\omega}$ we have

$$\frac{\partial(\nabla \cdot \tilde{\omega})}{\partial t} + \nabla \cdot (\tilde{u}(\nabla \cdot \tilde{\omega})) = 0,$$

as noted in [13]. The problem is common to all regularized vortex particle methods and work is still in progress to overcome this issue. An overview of further possible approaches tackling the violation of the divergence-free condition can be found in [13] and the references therein. However, an extensive discussion would exceed the scope of this work, so that we will not discuss this issue further and use the classical scheme for the time being.

The transpose and mixed schemes do not share this property, so that it seems natural to choose the classical scheme for an integration operator. We summarize the numerical vortex particle method in the concluding theorem.

Theorem 2.1.6
For suitable choices of initial particle positions $x_p(0)$, initial strength vectors $\alpha_p(0)$ and a smoothing function ζ, we obtain the following numerical scheme for a regularized vortex particle method in three-dimensional unbounded space

$$\begin{aligned}
\tilde{u}_\sigma(x,t) &= \sum_{p\in\mathcal{P}} K_\sigma(x - x_p(t)) \times \alpha_p(t), \\
\frac{\mathrm{d}x_p(t)}{\mathrm{d}t} &= \tilde{u}_\sigma(x_p(t),t), \\
\frac{\mathrm{d}\alpha_p(t)}{\mathrm{d}t} &= (\alpha_p(t)\cdot\nabla)\,\tilde{u}_\sigma(x_p(t),t) + \nu\Delta\alpha_p(x_p(t),t).
\end{aligned}$$

Remark 2.1.7
We note that the evolution equations of $\alpha_p(t)$ still include two differential operators, which have to be evaluated during numerical simulations as well. While the stretching term with its velocity derivatives can be derived analytically by differentiating the expression for $\tilde{u}_\sigma$, the viscous term has to be treated separately. One naive approach is to explicitly differentiate $\tilde{\omega}_\sigma$ (as done in Fishelov's method [48]), but this approach is known to be very sensitive to irregular particle positions [49] and does not conserve total vorticity [13]. We will return to a discussion of alternative viscous schemes in Section 2.2.3.

Computing $\tilde{u}_\sigma(x)$ for one particle $q \in \mathcal{P}$ involves the evaluation of $K_\sigma(x_q - x_p(t)) \times \alpha_p(t)$ for all $p \in \mathcal{P}$. Using $N = |\mathcal{P}|$ particles, the amount of operations required for the direct calculation of $\tilde{u}_\sigma(x)$ is of order $\mathcal{O}(N^2)$. Vortex particle methods are therefore one example of N-body problems, limiting the usability of a naive implementation to rather low N.

Although an implementation of the numerical scheme from Theorem 2.1.6 is already possible, it is mandatory to ask for the convergence properties of this approach. To this end, we state the central convergence result for inviscid vortex methods, i.e. for $\nu = 0$, thereby specifying, how 'suitable choices' for the initial conditions and the smoothing kernel should look like. This result has a significant impact on the design of regularized vortex particle methods.

2.1.4 Convergence for the inviscid case

The result presented in this section requires the usage of Sobolev spaces. The basic concept of these spaces can be found in [50, 51] and is cited in Appendix B. This ingredient is typical for convergence statements in the context of Navier-Stokes equations [7, 8] and will be necessary for our later discussion of the convergence analysis as well.

From what we already know, we can expect that convergence for the method stated in Theorem 2.1.6 fundamentally depends on the order of the smoothing function, its core size and the initial conditions. The next theorem gives the precise result [33].

Theorem 2.1.8 (Convergence for the inviscid case)
Let ζ be a smoothing function of order $r \geq 2$ with core size $\sigma > 0$ and

$$\zeta \in W^{m,\infty}(\mathbb{R}^3) \cap W^{m,1}(\mathbb{R}^3) \text{ for all } m \in \mathbb{N}.$$

We assume that for Theorem 2.1.6 the initial vorticity field $\omega(\cdot, 0)$ is smooth enough and that the initial particle positions $x(0)$ are separated by a sufficiently small value $h > 0$. Then there exists a time $T = T(\omega(\cdot, 0)) > 0$ and a constant $C = C(T) > 0$ with

$$\|(u - \tilde{u}_\sigma)(\cdot, t)\|_{L^p(\mathbb{R}^3)} \leq C\,\sigma^r \text{ for all } t \in [0, T],\ p \in (3/2, \infty]$$

if for $s, B > 0$ we have $h \leq B\,\sigma^{1+s}$.

Proof. A proof of this result can be found in [13, 33]. □

Remark 2.1.9

a) The imprecise requirement of the 'smoothness' of the initial vorticity field $\omega(\cdot, 0)$ is linked to the condition $h \leq B\,\sigma^{1+s}$. For $n > 0$ such that $h^n \leq \tilde{B}\,\sigma^{n+1}$, we require

$$\omega(\cdot, 0) \in W^{n+1,\infty}(\mathbb{R}^3) \cap W^{n+1,1}(\mathbb{R}^3),$$

as the proof shows [33]. This requirement is very similar to the one on the smoothing function and, indeed, it is because compact support for both vorticity field and smoothing functions is not mandatory here. While this is typically irrelevant for the vorticity field, requiring compact support would limit the choice of the smoothing function substantially.

b) Within the proof, the error induced by particle regularization and discretization is clearly split up. Using the same prerequisites, the proof exploits

$$\|(u - \tilde{u}_\sigma)(\cdot, t)\|_{L^p(\mathbb{R}^3)} \leq \|(u - u_\sigma)(\cdot, t)\|_{L^p(\mathbb{R}^3)} + \|(u_\sigma - \tilde{u}_\sigma)(\cdot, t)\|_{L^p(\mathbb{R}^3)},$$

where u_σ has to be interpreted as a regularized yet continuous representation of u. Now, the errors are given by

$$\begin{aligned} \|(u - u_\sigma)(\cdot, t)\|_{L^p(\mathbb{R}^3)} &\leq C_1\, \sigma^r, \\ \|(u_\sigma - \tilde{u}_\sigma)(\cdot, t)\|_{L^p(\mathbb{R}^3)} &\leq C_2\, h^m \sigma^{1-m}, \end{aligned}$$

with $m \in \mathbb{N}$ chosen so that $ms > r$. We note that the condition $h \leq B\,\sigma^{1+s}$ is not required for estimating the pure regularization error.

This convergence statement yields the basis and justification for regularized vortex particle methods. As expected, the order of the smoothing function and its core size determine the speed of convergence. But this is only valid if core size σ and initial inter-particle distance h are related by the so-called 'overlap criterion'

$$h \leq B\,\sigma^{1+s},$$

as depicted in Figure 2.2. Then, there exists a time $T > 0$, so that the approximation quality of $\tilde{u}_\sigma$ with respect to the original velocity field u as stated in Theorem 2.1.8 is guaranteed only for times $t < T$. After this critical, not explicitly known time T, the results of a numerical simulation should not be trusted anymore. Of course, this clearly prohibits long-term simulations and it is precisely the overlap criterion that necessitates further concepts for a stable numerical implementation. In the next section, we analyze the meaning and impact of this restriction and add viscous effects to the method.

2.2 Field distortions and viscous effects

For a numerical implementation of a stable version of the method proposed in Theorem 2.1.6, two open questions have to be addressed:

a) How can one perform reliable long-term simulations?

b) How can one treat viscous effects induced by the Laplacian of the vorticity field?

In this section, we present and discuss common approaches for these issues. Since the overlap criterion is already required for inviscid methods and has great impact on the accuracy of all regularized vortex particle methods, we start with an analysis of this restriction and point out possible remedies.

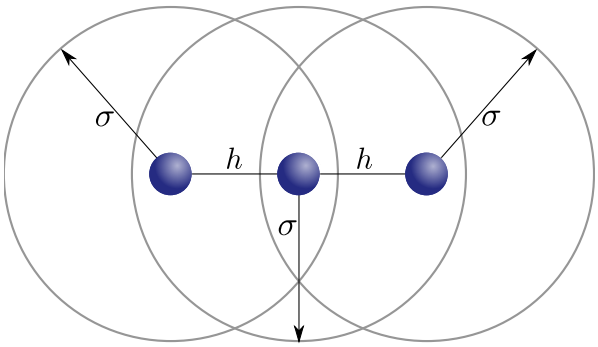


Figure 2.2: Three smoothed particles with distance h and core size $\sigma > h$: each particle overlaps with its nearest neighbors.

2.2.1 The overlap criterion

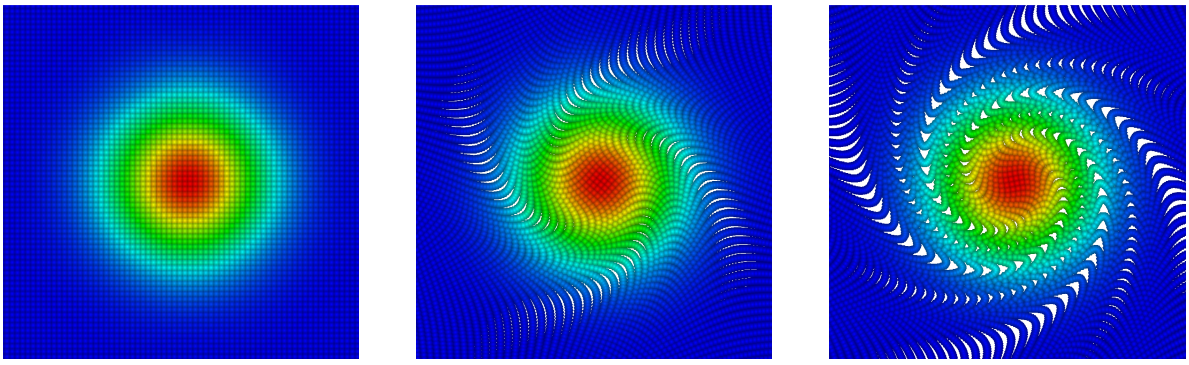
The overlap criterion couples inter-particle distances and core sizes, forcing the smoothed particles to overlap. Figure 2.2 depicts a very simple example: three smoothed particles with core size σ and distance h are arranged in a way, so that the particles are able to 'communicate' with each other [13]. This communication process is essential for the stability and accuracy of the method, as the Convergence Theorem shows. From the physical perspective, a flow may tend to distort the vorticity field, either unphysically clustering particles together or underresolving vortical structures where particles spread apart.

These phenomena can be observed easily by analyzing so-called 2D axisymmetric inviscid vortex patches [14]. In this commonly used two-dimensional setup, the initial vorticity field is discretized with particles located at grid points $x(0) = (i\,h, j\,h)$ for $i, j = 0, \ldots, \sqrt{N}$, scaled to the area $[-1, 1]^2$. Taking a radially symmetric vorticity field as initial condition, e.g.

$$\omega(x, 0) = (1 - |x|^2)^d \text{ for } d \in \mathbb{N} \text{ or}$$
$$\omega(x, 0) = \exp(-c\,|x|^2) \text{ for } c > 0,$$

the vorticity is in steady state, since no diffusion takes place and the vorticity is radially symmetric [52], so that we have $\omega(x, t) = \omega(x, 0)$ for all $x \in [-1, 1]^2$ and all $t \in [0, T]$. On the other hand, the particles move on circles, but with different speed. This effect leads to a rapid distortion of the vorticity field. Therefore, the 2D inviscid vortex patch is a very suitable benchmark and test case for regularized vortex particle methods, since it is very easy to set up and its result immediately reflects the compliance with the overlap criterion of a numerical implementation. There are various references using these setups with different strategies for the choice of σ:

- Perlman [44] and Choquin et al. [53] tested $\sigma = h^q$ for $q \in (0, 1)$,
- Barba et al. [54] chose $\sigma = c\sqrt{h}$ with $c \in (0.7, 0.9)$,
- Beale et al. [43] tried $\sigma = c\,h^{\frac{3}{4}}$ for e.g. $c \in \{1, 2, 2.5\}$.



(a) Time t = 0.0 **(b)** Time t = 5.0 **(c)** Time t = 10.0

Figure 2.3: 2D vortex patch with $N = 10000$ particles, zoomed into the core. Initial vorticity field $\omega(x, 0) = \exp(-12\,|x|^2)$, core size $\sigma = h^{0.95} = 0.02^{0.95}$. Third order Runge-Kutta time integration from $t = 0.0$ to $t = 10.0$ with step size $\Delta t = 0.1$. Vorticity magnitude decreases from the center to the outer areas. The three figures depict the initial conditions, as well as the times $t = 5.0$ and $t = 10.0$, where first severe disturbances occur, violating the initial symmetry of the vorticity field.

Tests in three dimensions are often much more complicated and the relationship between h and σ depends on case-specific geometric considerations [18, 55]. Therefore, the 2D vortex patch setup is a convenient choice for demonstrating the impact of the overlap criterion. Especially the symmetric character of the vorticity field should be conserved by a numerical scheme.

However, as Figure 2.3 shows this is anything but granted without further efforts. For the test case shown we use $\omega(x, 0) = \exp(-12\,|x|^2)$ and $h = 0.02$, resulting in $N = 10000$ particles initially located at grid points in $[-1, 1]^2$. We use a second order 2D Gaussian smoothing function with core size $\sigma = h^{0.95}$ (see [44] for details). A third order Runge-Kutta integration scheme [56] with time step $\Delta t = 0.1$ accounts for the evolution equations. It is obvious that after a short time the symmetry is disturbed by spurious artifacts due to the different speed of the particles, which leads to inaccurately resolved areas of vorticity. We analyze this example further at the end of this section. For now, we conclude that even for simple examples the Convergence Theorem is only of theoretical value, as long as the corresponding vortex particle algorithm does not provide a mechanism to accurately restore the vorticity field frequently.

There are various approaches available, which claim to prevent a numerical simulation from becoming unstable [13]. We briefly highlight three of them here: iterative circulation processing [57], rezoning [43, 58] and remeshing [59, 60].

Iterative circulation processing – This technique bases on the idea of constructing an interpolated vorticity field on the current particle locations out of the known vorticity values. This involves the solution of the system

$$A\beta = \alpha,$$

where the vectors α with $\alpha_p = \omega(x_p)\mathrm{vol}_p$ hold the 'old' vorticity values and the matrix A with elements $A_{p,q} := \zeta_\sigma(x_q - x_p) \cdot \mathrm{vol}_q$, $p, q \in \mathcal{P}$, represents the coefficients of the vorticity discretization. Then, the vector β contains the interpolated vorticity values at positions x_p. However, the matrix A is of size $N \times N$ if $|\mathcal{P}| = N$, so that for more than a few hundred particles A should not be inverted directly, especially since it is severely ill-conditioned in general [13]. As a workaround, an iterative process can be used, e.g.

$$\beta^{n+1} = \beta^n + (\beta^0 - A\beta^n)$$

with initial guess $\beta_p^0 = \omega(x_p)$ and a predefined $n \in \mathbb{N}$. However, the convergence of the iterative process is slowed down by strong overlapping of particles and lacking regularity of the vorticity field [13]. Therefore, this approach is often difficult to use in practical simulations.

Rezoning – The second method we mention here is the rezoning technique. After some time steps this scheme restarts the particles on a regular mesh with spacing h and attaches these new particles y_q, $q \in \mathcal{M}$,$|\mathcal{M}| = M$, with vorticity

$$\omega^{\mathrm{mesh}}(y_q) = \sum_{p \in \mathcal{P}} \zeta_\sigma(y_q - x_p) \cdot \omega^{\mathrm{part}}(x_p)\mathrm{vol}_p.$$

This approach originates from the vorticity field discretization as stated in Theorem 2.1.3, being a hybrid location and circulation processing scheme. Overlap can be guaranteed after restarting the simulation with the new particles, so that the Convergence Theorem holds again for a new time interval $[T, S]$. However, since most of the regularized smoothing kernels do not have compact support, a population control scheme has to be defined for the new particles, e.g. omitting particles with $|\omega^{\mathrm{mesh}}(y)| \leq \varepsilon$ for a predefined ε. Furthermore, a condition when to perform the rezoning procedure is necessary [58], especially since it most likely introduces unwanted diffusive errors [43], at least if it is performed with rather low order smoothing kernels [54]. We note that this scheme is of order $\mathcal{O}(NM)$ and therefore as expensive as the original vortex particle method of Theorem 2.1.6 (at least if M is of order N). Moreover, since the new particle locations are determined by using an underlying mesh structure, the initial mesh-free character of these methods is not fully conserved.

Remeshing – Finally, we review a pure location processing scheme. The remeshing approach, like rezoning, introduces a regular mesh with spacing h. The simulation is restarted after some time steps with new particles y_q, $q \in \mathcal{M}$, $|\mathcal{M}| = M$, at the mesh nodes, perfectly satisfying the overlap condition. As for rezoning, the Convergence Theorem is then applicable for a new time interval $[T, S]$. But instead of discretizing the vorticity field, remeshing

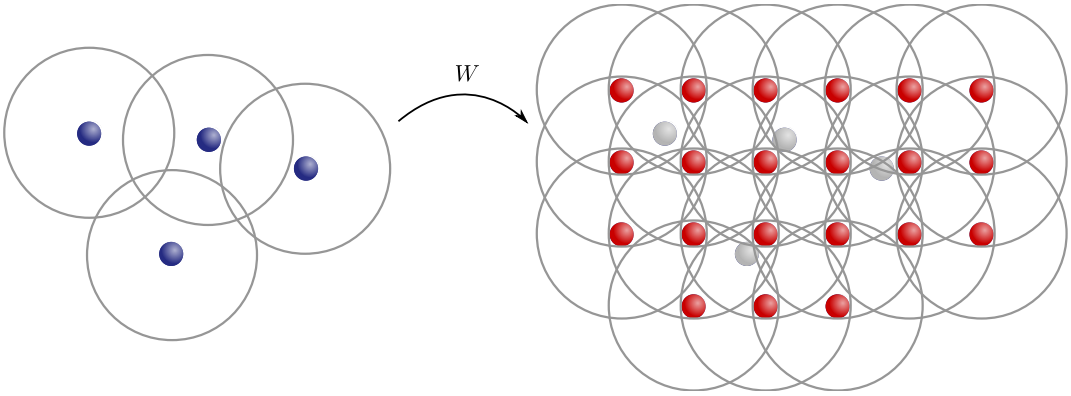


Figure 2.4: Four particles in two dimensions are projected onto a mesh by a kernel W with $3^2 = 9$ support nodes, yielding 21 new particles, which satisfy the overlap criterion by definition.

makes use of conservative interpolation kernels W, so that we have for the particle-to-mesh operator [52, 61]

$$\omega^{\text{mesh}}(y_q) = \sum_{p \in \mathcal{P}} W\left(\frac{|y_q - x_p|}{h}\right) \cdot \omega^{\text{part}}(x_p).$$

The properties of this relation are basically the same as for the rezoning scheme: we need to introduce a mesh structure for the restart procedure, population control is required and the computational cost for the naive implementation is of order $\mathcal{O}(NM)$. Moreover, we have to define when this scheme has to be applied (see [62] for a possible criterion). Despite these issues, this strategy is widely used in vortex particle methods [13], yielding a mostly non-diffusive means of ensuring the overlap condition [59]. The computational cost can even be reduced to $\mathcal{O}(N)$ by choosing an interpolation kernel W with compact support, making this scheme favorable.

The remeshing approach is schematically depicted in Figure 2.4 for four particles in two dimensions, lacking overlap, and a kernel W with a support of $3^2 = 9$ mesh nodes. The overlap criterion is clearly satisfied by definition, but instead of three particles the next step would contain 21 new particles. We examine the remeshing strategy further in the following and explain the derivation of the interpolation kernels in more detail.

2.2.2 The remeshing strategy for accurate vortex calculations

Performing a remeshing step prunes old particles x and restarts the simulation with new particles y located on a regular mesh. To ensure that during this procedure the characteristics of the system being simulated are conserved, we choose the interpolation kernel W to satisfy

specific moment conditions. For an arbitrary test function ϕ (e.g. the regularized smoothing kernel for the vorticity) in one spatial dimension we consider the error term

$$\varepsilon(x) := \phi(x) - \sum_{q\in\mathcal{M}} W\left(\frac{|y_q - x|}{h}\right) \cdot \phi(y_p).$$

Then the following theorem connects the moments conditions of W to the order of $\varepsilon(y)$.

Theorem 2.2.1
If for $m \in \mathbb{N}$ an interpolation kernel W satisfies the moment conditions

$$\sum_{q\in\mathcal{M}} W\left(\frac{|y_q - x|}{h}\right) \equiv 1,$$
$$\sum_{q\in\mathcal{M}} (y_q - x)^\beta \cdot W\left(\frac{|y_q - x|}{h}\right) \equiv 0 \text{ for all } 1 \leq |\beta| \leq m-1,$$

then for a constant $C > 0$ we have $|\varepsilon(x)| \leq C\, h^m$.

Proof. A deduction of these conditions can be found in [13]. □

Remark 2.2.2
These moment conditions are basically a discrete analog to the moment conditions of Definition 2.1.1 for the smoothing kernel. They express the conservation of total vorticity, linear, angular and higher momenta as well. Their analysis is performed best in Fourier space and we refer to [13, 59, 61] for details.

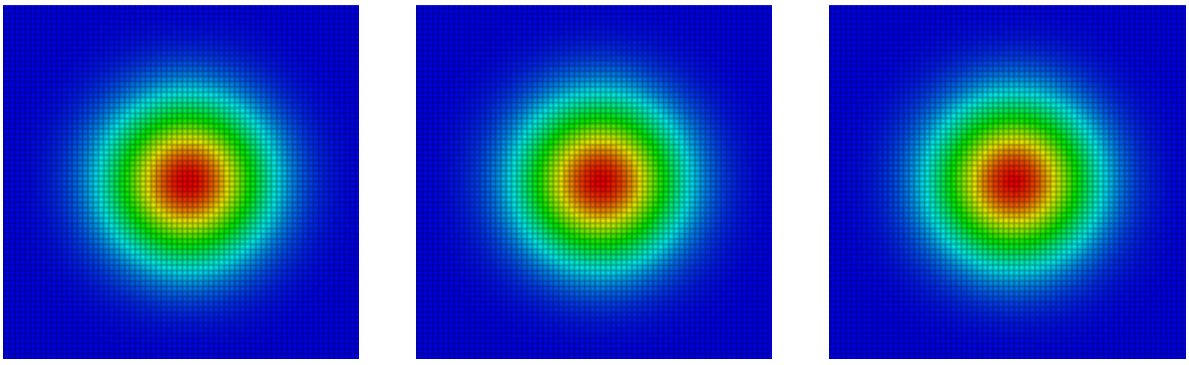
There are basically two approaches for choosing W: 'discrete' and 'smooth' interpolation. Kernels belonging to the first type can be directly deduced from the moment conditions of Theorem 2.2.1, yielding e.g. the linear cloud-in-cell (CIC) kernel [63, 64]

$$\Lambda_1(z) = \begin{cases} 1-z, & 0 \leq z \leq 1, \\ 0, & \text{otherwise.} \end{cases}$$

Another example is the piecewise cubic kernel

$$\Lambda_3(z) = \begin{cases} \frac{1}{2}(1-z^2)(2-z), & 0 \leq z < 1, \\ \frac{1}{6}(1-z)(2-z)(3-z), & 1 \leq z < 2, \\ 0, & \text{otherwise,} \end{cases}$$

which conserves the four first momenta [59, 65]. Its deduction from the moment conditions of Theorem 2.2.1 is reconstructed in Section 2.2.4. However, these kind of kernels do not



(a) Time t = 0.0 **(b)** Time t = 5.0 **(c)** Time t = 10.0

Figure 2.5: Again, 2D vortex patch with initially $N = 10000$ particles, zoomed into the core. Initial vorticity field $\omega(x, 0) = e^{-12|x|^2}$, core size $\sigma = h^{0.95} = 0.02^{0.95}$. Third order Runge-Kutta time integration from $t = 0.0$ to $t = 10.0$ with step size $\Delta t = 0.1$, remeshing every fifth time step. Vorticity magnitude decreases from the center to the outer areas. The three figures depict the initial conditions and times $t = 5.0$ and $t = 10.0$. The distortions of Figure 2.3 have vanished, the initial symmetry of the vorticity field is conserved.

have continuous derivatives and higher order kernels may not be even continuous themselves. Relaxing the condition $\tilde{\omega}_\sigma^{\text{mesh}}(y) = \tilde{\omega}_\sigma^{\text{part}}(x)$ for $y = x$, we can construct kernels with smooth derivatives by using central B-splines [60]. One widely used formula is the M_4'-kernel [66]

$$M_4'(z) = \begin{cases} 1 - \frac{5}{2}z^2 + \frac{3}{2}z^3, & 0 \leq z < 1, \\ \frac{1}{2}(2 - z)^2(1 - z), & 1 \leq z \leq 2, \\ 0, & \text{otherwise}, \end{cases}$$

which reproduces polynomials up to order two and therefore conserves total vorticity, linear and angular momentum. It has compact support with extension of four mesh nodes (like the Λ_3-kernel), being therefore a suitable kernel for $\mathcal{O}(N)$ remeshing methods. Using simple tensor product formulas, we can use this kernel (and others as well) for remeshing in more than one dimension, so that the M_4'-kernel uses 4^d mesh nodes in d spatial dimensions.

This kernel is widely used in other simulations – even for smoothed particle hydrodynamics, as described in [67] – and it is common to perform remeshing every first to every tenth time step, while population control is handled either by relative or absolute thresholds with respect to vorticity magnitude [54, 65, 67, 68, 69]. We adopt these strategies and return to the initial example of Figure 2.3. In Figure 2.5, the same setup as before is shown, but now with remeshing applied every fifth time step and corresponding rigorous population control leading to $N = 9372$ particles at $t = 10.0$, using an absolute threshold for the vorticity of 10^{-6}. The vorticity is perfectly resolved and no artifacts disturb the initial symmetry. In Figure 2.6, the

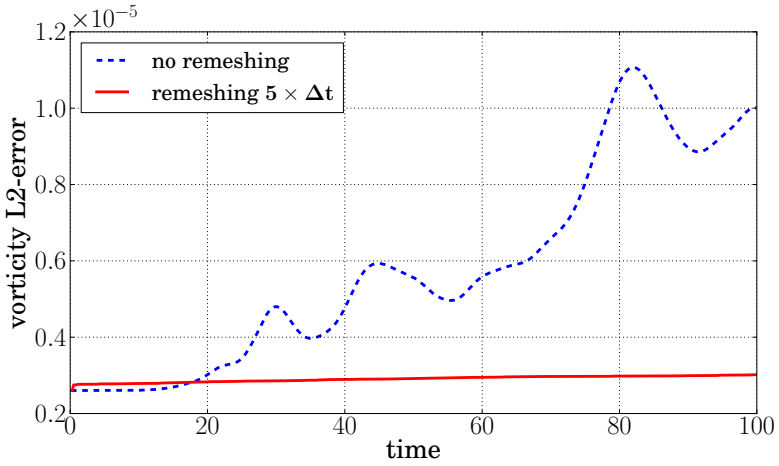


Figure 2.6: Discrete L^2-error norms $\|u - \tilde{u}_\sigma\|_2$ of the velocity, same data as Figure 2.5. No remeshing vs. remeshing every fifth time step. $N = 10000$ to $N = 8970$ at $t = 100.0$ with absolute population control threshold of 10^{-6} for the vorticity.

computed velocities $\tilde{u}_\sigma$ and the analytically known expression

$$u(x) = \frac{1}{24|x|^2}\left(1 - e^{-12|x|^2}\right)\begin{pmatrix} -x_2 \\ x_1 \end{pmatrix}$$

are compared using the discrete L^2-error norm

$$\|u - \tilde{u}_\sigma\|_2 := \frac{1}{|\mathcal{P}|}\left(\sum_{p\in\mathcal{P}} |u(x_p) - \tilde{u}_\sigma(x_p)|^2\right)^{\frac{1}{2}}.$$

While the error plot for the original method behaves very unstable after $t = 20.0$, thereby reflecting the resolution problems of the underlying vorticity fields, the remeshed method remains numerically stable with an L^2-error one order of magnitude smaller at the end of the simulation. As noted in [54], most of the accuracy is lost during the first remeshing step and until $t = 18.0$ the non-remeshed simulation performs slightly better. However, even for these simple setups the remeshing scheme performs very well in the long run, preventing particle distribution distortion and therefore conserving the required overlap between neighboring particles.

Since the remeshing procedure makes use of an underlying mesh structure, the initial mesh-free character of vortex particle method is somewhat compromised. There is one approach that exploits precisely this fact: the vortex-in-cell algorithm. In the style of particle-in-cell or cloud-in-cell methods, this scheme projects particles onto the mesh and uses fast Poisson solvers and finite differences for computing the fields and derivatives on the mesh. Afterwards, the same interpolation kernel is used for projecting the fields back onto the particles,

which are then moved by the integrator. Since this approach has to be counted among mesh-based methods, we will not discuss it here. For more details we refer to [13, 70].

As we have seen in this section, the overlap criterion is crucial not only for theoretical reasons, but even for practical yet simple simulations. A location or circulation processing scheme like remeshing is therefore mandatory. The overlap of nearby particles – i.e. the ability to communicate – is required for viscous flows as well. In the next section, we present two fundamentally different approaches tackling the diffusive part of vortex particles methods, the second one eventually exploiting the remeshing approach further to directly account for viscous effects.

2.2.3 The diffusion term

We now focus on viscous scenarios with $\nu > 0$. In this case the diffusive term $\nu\,\Delta\alpha_p$ or, to simplify matters, $\nu\Delta\omega$ necessitates a numerical treatment of the Laplacian Δ. There are many different approaches available, e.g. particle strength exchange (PSE) [71], explicit differentiation [48], random-walk methods [40] and vortex redistribution methods (VRM) [72]. A very comprehensive overview on viscous vortex methods can be found in [49, 54].

Firstly, we review the first two techniques, since the PSE-scheme is widely used [69, 73] and has a consistent convergence statement [71], while the explicit differentiation approach can be seen as modification of this method. Furthermore, some of the results derived later in Chapter 3 can be transferred to these methods.

In short, the key concepts behind PSE are the approximation of the Laplacian using an integral operator and the discretization of this operator using particles. For the first step, we define the approximation of the diffusion operator by

$$\Delta_\sigma\omega(x) := \frac{1}{\sigma^2}\int_{\mathbb{R}^3} \eta_\sigma(y-x)\cdot(\omega(y)-\omega(x))\mathrm{d}y$$

with a regularized approximation kernel

$$\eta_\sigma(x) := \frac{1}{\sigma^3}\,\eta\left(\frac{x}{\sigma}\right),$$

where we require $\eta \in L^1(\mathbb{R}^3)$ and $\eta(x) = \eta(-x)$. This definition of the regularized approximation kernel is reminiscent of Definition 2.1.1 of smoothing kernels ζ and indeed, approximation kernels η are often constructed by using [71]

$$\eta(x) = -2\,\frac{x\nabla\zeta(x)}{|x|^2}.$$

For a convergence statement we would expect to satisfy certain moment conditions for these kernels. These conditions look very similar to the ones from Definition 2.1.1 and read

$$\int_{\mathbb{R}^3} \eta(y)\, y^\beta \mathrm{d}y = \begin{cases} 2, & \text{if } \beta \in \{(2,0,0),(0,2,0),(0,0,2)\}, \\ 0, & \text{else and } 1 \leq |\beta| \leq r+1, \end{cases}$$
$$\int_{\mathbb{R}^3} |\eta(y)|\, |y|^{r+2} \mathrm{d}y < \infty$$

for a multi-index β (see Definition A.3 for details on multi-index notation). Using these moment conditions the PSE approach is conservative up to order r. The second step consists of the numerical integration of $\Delta_\sigma \omega$, i.e. the discretization of the integral with regularized particles. We use

$$\tilde{\Delta}_\sigma \omega(x) := \frac{1}{\sigma^2} \sum_{q \in \mathcal{P}'} \eta_\sigma(x - x_q) \cdot (\omega(x_q) - \omega(x)) \mathrm{vol}_q$$

for quadrature points x_q with volume vol_q, $q \in \mathcal{P}'$, which are in most cases the simulation particles themselves.

On the other hand, explicit differentiation can be seen as a special version of PSE by choosing

$$\eta(x) = \Delta\zeta.$$

But while an explicit differentiation approach, as done by Fishelov in [48], would yield

$$\Delta\omega(x) \approx \frac{1}{\sigma^2} \sum_{q \in \mathcal{P}'} \Delta\zeta_\sigma(x - x_q) \cdot \omega(x_q) \mathrm{vol}_q,$$

the PSE deduction states the conservative form of this approximation with

$$\Delta\omega(x) \approx \frac{1}{\sigma^2} \sum_{q \in \mathcal{P}'} \Delta\zeta_\sigma(x - x_q) \cdot (\omega(x_q) - \omega(x)) \mathrm{vol}_q,$$

rendering explicit differentiation unfavorable.

Analyzing the convergence properties of approximation and regularized discretization reveals that the overlap criterion plays a crucial role also in the PSE scheme [71]. This is not surprising, since the discretization of the integral has already been the reason for the emergence of the overlap condition in the context of inviscid vortex particle methods, as noted in Remark 2.1.9. Of course, remeshing can be used here as well to account for distorted vorticity fields, so that the overlap criterion poses no additional limitations to this method. We can therefore conclude that PSE can be seen as a natural extension of inviscid regularized vortex particle methods. The choice of the approximation kernel η can be coupled to the smoothing kernel ζ and the convergence depends for both schemes on the particle distribution. We note that

since the calculation of $\tilde{\Delta}_\sigma \omega(x_p)$ for one particle x_p, $p \in \mathcal{P}$, involves the summation over $M := |\mathcal{P}'|$ quadrature points, the PSE scheme requires $\mathcal{O}(NM)$ operations.

While the PSE scheme can directly operate on the vorticity-update term, other viscous methods require a clear distinction between the inviscid and viscous parts of the governing equations. This so-called viscous splitting approach separates the convective and diffusive effects into two successive sub-steps [40], which can be applied independently from each other. In more detail, these steps can be written as (see Section 2.1.3 for notation)

a) the convection equation

$$\begin{aligned} \frac{\mathrm{d}x_p(t)}{\mathrm{d}t} &= \tilde{u}_\sigma(x_p(t), t), \\ \frac{\mathrm{d}\alpha_p(t)}{\mathrm{d}t} &= 0, \end{aligned}$$

b) the diffusion equation

$$\begin{aligned} \frac{\mathrm{d}x_p(t)}{\mathrm{d}t} &= 0, \\ \frac{\mathrm{d}\alpha_p(t)}{\mathrm{d}t} &= \nu\, \Delta\alpha_p(x_p(t), t). \end{aligned}$$

In multi-step methods the convection process is repeated multiple times before viscous effects are applied. With this separation at hand, an algorithm treating the viscous term can focus on the pure diffusion equation. However, an additional error of order $\mathcal{O}(\nu \Delta t_D)$ is induced [13], where Δt_D is the time step between two diffusive sub-steps.

One method exploiting viscous splitting is the so-called vortex redistribution method [72, 74, 75]. In this method, the vorticity field is updated by the redistribution of fractions of strength between neighboring particles, i.e. for a given vorticity ω^n at time step n with

$$\omega^n(x) = \sum_{p \in \mathcal{P}} \zeta_\sigma(x - x_p) \cdot \alpha_p^n$$

we obtain after the diffusion sub-step

$$\omega^{n+1}(x) = \sum_{p \in \mathcal{P}} \sum_{q \in \mathcal{P}'} \zeta_\sigma(x - x_q) \cdot f_{p,q}^n \cdot \alpha_p^n.$$

The neighborhood $\mathcal{P}'$ of a particle x is initially defined as all particles y with

$$|x - y| \leq R\, h_\nu := R \sqrt{\nu\, \Delta t},$$

where R is a predefined constant (typically $R = \sqrt{12}$) and h_ν is the so-called 'diffusion distance' [74]. Again, the fractions $f_{p,q}^n$ are defined by a set of familiar moment conditions

$$\sum_{q \in \mathcal{P}'} f_{p,q}^n = 1,$$

$$\sum_{q \in \mathcal{P}'} f_{p,q}^n \cdot (x_p - x_q)^\beta = \begin{cases} 2h_\nu^2, & \text{if } \beta \in \{(2,0,0),(0,2,0),(0,0,2)\}, \\ 0, & \text{else and } 1 \leq |\beta| \leq r+1, \end{cases}$$

which are basically a discrete version of the PSE moment conditions, with a small change for the diffusive contribution. They have to be satisfied at least up to order $r = 1$ to obtain consistency of the method [54, 72]. The fractions $f_{p,q}^n$ should be determined explicitly by these conditions, but depending on the regularity of the particles field the neighborhood $\mathcal{P}'$ of a particle might not contain enough target particles. In this case, additional particles are inserted to compensate for this problem.

The main drawback of this method is its dependency on a fast solution for the moment conditions. Since these equations depend on the actual particle position x_p, they have to be solved for each particle and each time step to compute the new fractions. The insertion procedure is indeed an intrinsic mesh-free approach, but on the other hand the overlap criterion is not guaranteed to be satisfied, although it is of course still required for the inviscid part [54]. Therefore, remeshing is required again and it turns out that this fact can be even of great help: the remeshing kernels can be rewritten so that they are able to account for viscous effects and field distortion simultaneously.

2.2.4 Modified remeshing and viscous effects

In contrast to the original vorticity redistribution method, the following modification changes particle locations and quantities during redistribution. If this process is done for a regular mesh of target particles, the solution of the equation systems for the fractions becomes much easier and can even be integrated into existing remeshing kernels. This 'modified remeshing' approach has been introduced in [76] and we review their ideas in this section. The basis for this approach is the observation that the moment conditions for remeshing kernels and redistribution fractions are nearly identical. Therefore, instead of using the conditions of Theorem 2.2.1 for constructing remeshing kernels, we use the redistribution moment conditions, which vary only for $\beta = (2,0,0),(0,2,0),(0,0,2)$, so that the resulting kernels can satisfy both remeshing and redistribution conditions.

To clarify this approach we derive the modified Λ_3-kernel, assuming a configuration in one spatial dimension [76]. Since Λ_3 extends to four mesh particles, we define

$$x_1 := -\frac{3}{2}h, \quad x_2 := -\frac{1}{2}h, \quad x_3 := \frac{1}{2}h \quad \text{and} \quad x_4 := \frac{3}{2}h.$$

Furthermore, we consider a particle x_0 with $|x_0| < \frac{1}{2}h$ and without loss of generality we assume $x_0 \in [0, \frac{1}{2}h)$. We obtain

$$\frac{3}{2} \leq \frac{|x_0 - x_1|}{h} = \frac{x_0}{h} + \frac{3}{2} < 2$$

and the solution W of the set of equations

$$\begin{aligned}
\sum_{j=1}^{4} W\left(\frac{|x_0 - x_j|}{h}\right) &= 1, \\
\sum_{j=1}^{4} (x_0 - x_j) \cdot W\left(\frac{|x_0 - x_j|}{h}\right) &= 0, \\
\sum_{j=1}^{4} (x_0 - x_j)^2 \cdot W\left(\frac{|x_0 - x_j|}{h}\right) &= 0, \\
\sum_{j=1}^{4} (x_0 - x_j)^3 \cdot W\left(\frac{|x_0 - x_j|}{h}\right) &= 0
\end{aligned}$$

yields for $z = \frac{|x_0 - x_1|}{h} \in [\frac{3}{2}, 2)$

$$W(z) = \frac{1}{6}(1 - z)(2 - z)(3 - z),$$

reproducing the correct branch of the Λ_3-kernel. Replacing the third equation with

$$\sum_{j=1}^{4} (x_0 - x_j)^2 \cdot W\left(\frac{|x_0 - x_j|}{h}\right) = 2\nu\,\Delta t,$$

we obtain

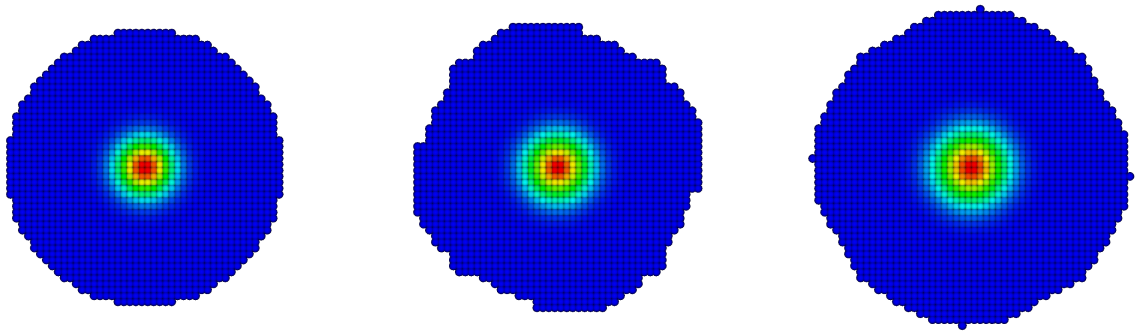
$$W(z) = \frac{1}{6}(1 - z)(2 - z)(3 - z) + c^2(2 - z)$$

for

$$c^2 = \frac{\nu\,\Delta t}{h^2}.$$

Using x_2, x_3 and x_4 in z we finally obtain the modified Λ_3-kernel

$$\Lambda_3(z, c) = \begin{cases} \frac{1}{2}(1 - z^2)(2 - z) - c^2(2 - 3z), & 0 \leq z < 1, \\ \frac{1}{6}(1 - z)(2 - z)(3 - z) + c^2(2 - z), & 1 \leq z < 2, \\ 0, & \text{otherwise.} \end{cases}$$



(a) Time t = 4.02 **(b)** Time t = 5.00 **(c)** Time t = 6.00

Figure 2.7: Lamb-Oseen vortex with initially $N = 2500$ particles, located in $[-0.5, 0.5]^2$, core size $\sigma = h^{0.95} = 0.02^{0.95}$, viscosity $\nu = 5 \cdot 10^{-4}$. Third order Runge-Kutta time integration from $t = 4.0$ to $t = 6.0$ with step size $\Delta t = 0.01$. Vorticity magnitude decreases from the center to the outer areas. Remeshing and diffusion performed in each time step, leading to $N = 2112$ particles at $t = 6.0$. The three figures depict the field after the first time step, at times $t = 5.0$ and $t = 6.0$. Initial conditions are not shown to emphasize the growth of the vorticity support.

Similarly, the M_4'-kernel can be extended to [76]

$$M_4'(z,c) = \begin{cases} 1 - \frac{5}{2}z^2 + \frac{3}{2}z^3 - c^2(2 - 9z^2 + 6z^3), & 0 \leq z < 1, \\ \frac{1}{2}(2-z)^2(1-z) - c^2(2-z)^2(1-2z), & 1 \leq z \leq 2, \\ 0, & \text{otherwise.} \end{cases}$$

Stability considerations in Fourier space [76] for the parameter c and the kernel $M_4'(z,c)$ require

$$\frac{1}{6} \leq c^2 \leq \frac{1}{2},$$

yielding e.g. non-negativity of the kernel.

To verify this approach we simulate the so-called Lamb-Oseen vortex [77] with vorticity field

$$\omega(x,t) = \frac{1}{4\pi\nu t} \exp\left(-\frac{|x|^2}{4\nu t}\right),$$

starting from an initial vortex patch in $[-0.5, 0.5]^2$ with $N = 2500$ particles and $h = 0.02$. Again, as for the remeshing test in Figure 2.5, we apply a third order Runge-Kutta integrator, now from $t = 4.0$ to $t = 6.0$ with step size $\Delta t = 0.01$. The viscosity is chosen as $\nu = 5 \cdot 10^{-4}$. Using the modified remeshing scheme with diffusion in every time step, we end up with $N = 2112$ particles at time $t = 6.0$. Figure 2.7 indicates that the support of the vorticity field

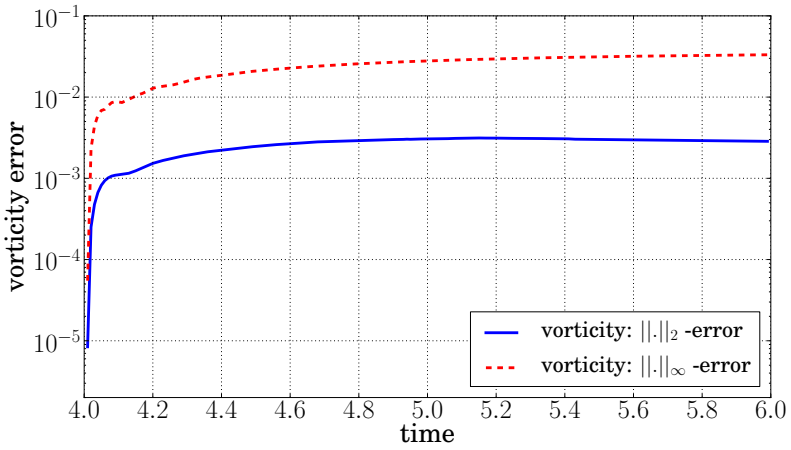


Figure 2.8: Error plot for the Lamb-Oseen vortex with initially $N = 2500$ particles, located in $[-0.5, 0.5]^2$. Core size $\sigma = h^{0.95} = 0.02^{0.95}$, viscosity $\nu = 5 \times 10^{-4}$. Third order Runge-Kutta time integration from $t = 4.0$ to $t = 6.0$ with step size $\Delta t = 0.01$. Remeshing and diffusion performed in each time step, leading to $N = 2112$ particles at $t = 6.0$.

increases significantly, as expected. The errors of Figure 2.8 are computed by the discrete L^2-error

$$\|\omega^{\text{exact}} - \omega^{\text{comp}}\|_2 := \frac{1}{|\mathcal{P}|} \left(\sum_{p \in \mathcal{P}} |\omega^{\text{exact}}(x_p) - \omega^{\text{comp}}(x_p)|^2 \right)^{\frac{1}{2}}$$

and the relative maximum error

$$\|\omega^{\text{exact}} - \omega^{\text{comp}}\|_\infty := \frac{\max\{|\omega^{\text{exact}}(x_p) - \omega^{\text{comp}}(x_p)|, p \in \mathcal{P}\}}{\max\{|\omega^{\text{exact}}(x_p)|, p \in \mathcal{P}\}},$$

respectively. The results correspond very well to those given in [49, 54]. The discrete L^2-error and the relative maximum error immediately start to grow, evening out in the range of $2{\cdot}10^{-3}$ and $2{\cdot}10^{-2}$, respectively. The maximum vorticity (scalar in two dimensions) decreases according to the exact values of the Lamb-Oseen vortex and the errors of Figure 2.8.

Despite the tempting rigorous mathematical background of the PSE method, the modified vorticity redistribution method shows its strength in the implementation. With this approach, every inviscid algorithm which has already implemented remeshing with an interpolation kernel W can be easily extended to handle viscous effects using a modified kernel $\hat{W}$ without computational overhead. However, general convergence results are yet unknown. Only considerations for the classical heat equation have been made so far, yielding an error of $\mathcal{O}((v\Delta t)^{\frac{r}{2}})$ in addition to the viscous splitting error [72, 76].

2.3 Summary

In this chapter, we have seen how the vorticity-velocity equation of Definition 1.2.2 and the Biot-Savart kernel of Theorem 1.2.5 can be used to derive a particle method simulating vorticity-driven flows. Figure 2.9 updates the introductory choice between mesh-based and particle-based methods in favor of the latter approach. The mathematical consequences and implications of this decision have been the topic of this chapter.

We have seen that the introduction of regularized smoothing kernels paves the way for a convenient discretization and a consistent convergence analysis for the inviscid case. In contrast to the point vortex method, higher order regularization yields a better speed of convergence, making this approach favorable for lengthy simulations. Momenta conservation and the regularity of the underlying particle field play crucial roles for the convergence as well. The central requirement for convergence – the overlap criterion – couples inter-particle distance and the core radius of each particle. For long-term simulations, this has to be maintained by a numerical implementation. Besides many other approaches, remeshing proved to be a very effective and natural way of restoring the convergence condition. Although introducing a mesh-like structure, the conservation properties of the interpolation kernels – again controlled by moment conditions – and its simplicity make this tool the first choice for most simulations. Moreover, we have seen that the remeshing approach can be exploited even further to account not only for ensuring long-term reliability of the method, but also for viscous effects by introducing modified redistribution kernels.

We combine the results of this chapter into the following algorithm, stating the workflow of a numerical vortex particle method handling convection, stretching, diffusion, remeshing and time integration. We choose a time $T > 0$, initial vorticity $\omega(\cdot, 0)$ and initial particle locations $x_p(0)$, $p \in \mathcal{P}$ with distances h and core sizes σ. Furthermore, a specific viscous vortex particle method is defined by the choices of the smoothing kernel ζ, the modified interpolation kernel $\hat{W}$ and the time integration scheme. The workflow can be written in pseudocode as depicted in Algorithm 2.1.

With small additional modifications for the '$\leftarrow$'-assignments, this algorithm yields the main loop of a regularized vortex particle method and its subroutines COMPUTEINTERACTION and MODREMESHING. The modified remeshing and diffusion operator of MODREMESHING can

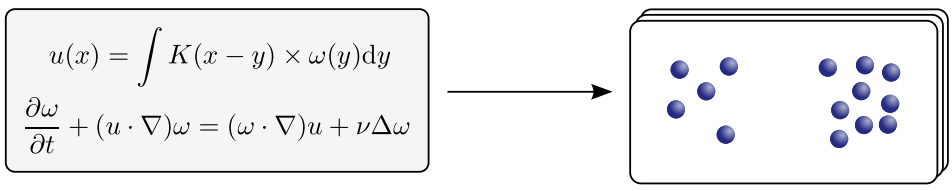


Figure 2.9: The transition from analytic equations to particle-based methods.

be implemented efficiently using only $\mathcal{O}(N)$ operations with $N = |\mathcal{P}|$. In Chapter 4, one implementation using an adaptive, particle-only approach with parallel sorting is described. However, in COMPUTEINTERACTION the calculation of $\tilde{u}_\sigma$ and its derivative requires $\mathcal{O}(N^2)$ operations, limiting the usability to rather small numbers of N. Since all relevant kernels – the regularized smoothing function ζ_σ, the stream function kernel G_σ and its derivative K_σ – typically lack compact support, their influence even on distant particles has to be considered carefully. Vortex particle methods are therefore an example of N-body problems dominated by long-range interactions. We note that the same problem applies for particle strength exchange and rezoning if these methods are used with non-compact kernels to replace diffusion and remeshing (see Section 2.2). Therefore, the goal of the next chapter is the formulation of a fast summation method to replace COMPUTEINTERACTION, making simulations with million of particles possible in a reasonable time scale.

Algorithm 2.1 Workflow of a vortex particle method

subroutine COMPUTEINTERACTION ▷ Compute velocities and vorticity updates
  $\tilde{u}_\sigma \leftarrow 0$
  for all $p \in \mathcal{P}$ **do**
    for all $q \in \mathcal{P}$ **do**
      $\tilde{u}_\sigma(x_p) \leftarrow \tilde{u}_\sigma(x_p) + K_\sigma(x_p - x_q) \times \alpha_q$
    end for
  end for
  $\alpha_p^{\text{update}} \leftarrow (\alpha_p \cdot \nabla)\tilde{u}_\sigma(x_p)$
end subroutine

subroutine MODREMESHING ▷ Modified remeshing and diffusion
  for all $m \in \mathcal{M}$ **do**
    $\alpha_m^{\text{mesh}} \leftarrow 0$
    for all $p \in \mathcal{P}$ **do**
      $\alpha_m^{\text{mesh}} \leftarrow \alpha_m^{\text{mesh}} + \hat{W}\left(\frac{|x_m^{\text{mesh}} - x_p|}{h}\right) \cdot \alpha_p$
    end for
  end for
  $\alpha_p \leftarrow \alpha_m^{\text{mesh}}$, $x_p \leftarrow x_m^{\text{mesh}}$, $\mathcal{P} \leftarrow \mathcal{M}$
end subroutine

program MAIN ▷ Main loop
  while $t < T$ **do**
    call COMPUTEINTERACTION
    for all $p \in \mathcal{P}$ **do**
      $x_p^{\text{new}} \leftarrow \frac{\mathrm{d}x_p}{\mathrm{d}t} = \tilde{u}_\sigma(x_p)$
      $\alpha_p^{\text{new}} \leftarrow \frac{\mathrm{d}\alpha_p}{\mathrm{d}t} = \alpha_p^{\text{update}}$
    end for
    $t \leftarrow t + \Delta t$
    if remeshing and diffusion requested **then**
      call MODREMESHING
    end if
  end while
end program

3 Multipole expansions and a class of algebraic kernels

We have seen in the last chapter that using the mesh-free vortex particle method results in an N-body problems for the evaluation of $\tilde{u}_\sigma$ and its derivatives. Despite the relentless technological advances in supercomputing performance over the past decade, solving the N-body problem for such systems dominated by long-range interactions remains a formidable algorithmic and computational challenge. The brute-force approach, in which all N^2 mutual interactions between particles are computed directly, inevitably becomes impractical for large systems. However, this approach has a straightforward parallelization scheme, which allows for effective distribution of the particle set across large machines. But even exploiting fast compute nodes and distributed-memory architectures does not avert the inherent $\mathcal{O}(N^2)$-complexity, making dynamic simulations with million of particles economically impractical. Figure 3.1 illustrates two fundamentally different approaches, which aim at accelerating N-body simulations while maintaining the mesh-free character: cutoff-based or multipole-based methods. Again, we highlight and briefly weigh pros and cons of both concepts in the following.

Cutoff-based methods – Cutoff-based methods for long-range particle-particle interactions are commonly used in molecular dynamics [78, 79, 80]. They rely on choosing a cutoff radius ρ_{cut}, which trims the prevailing interactions to zero at $\rho = \rho_{\text{cut}}$, thus omitting direct particle interactions outside a sphere with radius ρ_{cut}. Consequently, for constant radii and homo-

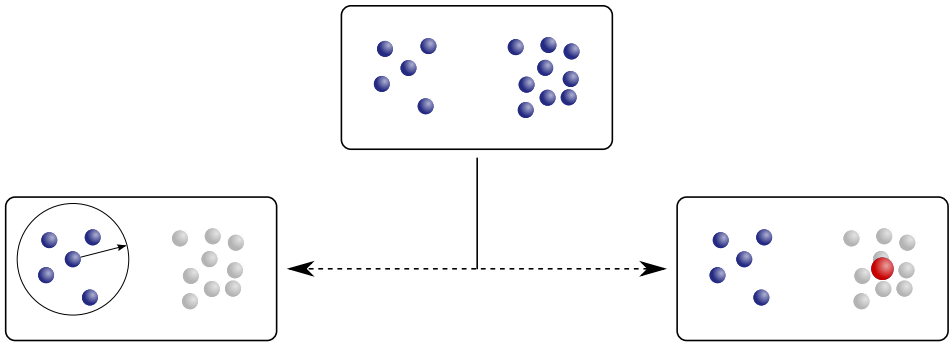


Figure 3.1: Computation of interactions using cutoff-based or multipole-based methods?

geneous distributions only a constant number of particles are required for the summation. Methods like the direct Wolf summation can be implemented and parallelized easily with a algorithmic scaling of $\mathcal{O}(N)$, but they are often motivated and justified only by physical considerations of the system of interest [80, 81].

Multipole-based methods – Two multipole-based techniques developed in the mid-1980s – the hierarchical Barnes-Hut tree code [82] and the Fast Multipole Method (FMM) [83], with respective algorithmic scalings of $\mathcal{O}(N \log N)$ and $\mathcal{O}(N)$ – have become widely used mesh-free means for long-range N-body simulations across a broad range of fields [84, 85, 86, 87, 88]. Both methods reduce the number of direct particle-particle interactions through the use of so-called multipole expansions, making it possible to perform simulations with millions of particles. The summation for each particle is computed by systematically replacing more distant particles by multipole expansions of groups, thus reducing the standard $\mathcal{O}(N^2)$ direct sum to an $\mathcal{O}(N \log N)$- or even $\mathcal{O}(N)$-complexity at the price of a adjustable and controllable error [89].

Both approaches preserve the mesh-free character of particle simulations. While the mathematical background and practical implementation of multipole-based methods are rather complex, cutoff-based methods allow for a straightforward access. Conversely, this apparent advantage comes at the cost of mathematical drawbacks. In many situations the convergence analysis of cutoff-based methods is not rigorous but often based on heuristic observations or physical considerations, which limit their application to very specific cases [81]. On the other hand, multipole-based methods like the Barnes-Hut tree code or the Fast Multipole Method rely on well-grounded, consistent and generic mathematical theories [83, 90, 91]. Their algorithmic complexity can be easily estimated, but their implementation and parallelization on distributed machines is not straightforward. Both multipole-based concepts share the abstract idea of multipole expansions and have to be modified only for varying interaction kernels, but not for different physical setups. Besides the consistent mathematical background, this property makes multipole-based methods favorable in our case. In terms of vortex particle methods, a rigorous mathematical analysis of multipole expansions and their convergence properties has to start from an explicit choice of smoothing kernels.

The goal of this chapter is the adaption of the concept of multipole expansions for the case of regularized vortex particle methods using algebraic smoothing kernels, paving the way for rapid numerical simulations with millions of vortex particles. In terms of Algorithm 2.1 of Section 2.3 the upcoming chapter constitutes the mathematical basis for replacing the $\mathcal{O}(N^2)$-subroutine COMPUTEINTERACTION with a much faster, multipole-based equivalent.

To this end, we will introduce a novel and comprehensive class of algebraic smoothing kernels, which utilizes and extends the well-known zeroth and second order algebraic kernels of Section 2.1.2. Using the moment conditions of Definition 2.1.1, we will show how to derive kernels of arbitrary order, each remaining within the framework of the class. A decomposition theorem will prove to be very helpful for the analysis of the class and its members, especially concerning their usability in the context of multipole-based methods. After a short review of

the mathematics of general multipole expansion theory we will apply this formalism to our class by means of the decomposition theorem. Starting from the stream function smoothing kernel G_σ we will deduce a comprehensive theory of multipole expansions in the field of vortex particle methods with algebraic regularization. Error estimation and convergence result will turn out to nicely extend well-known results for common ρ^{-n}-kernels and can even be applied to particle strength exchange and rezoning. An extension to field derivatives, thus covering $\tilde{u}_\sigma$ and its derivatives, will conclude this chapter.

3.1 A class of algebraic kernels

As we have seen in the previous chapter, vortex particle methods are primarily defined by the choice of the smoothing kernel. This function determines the vorticity discretization, the stream function and – by applying multiple differential operators – the velocity evaluation, as well as the vorticity update in 3D. Therefore, a complete multipole expansion theory for given smoothing kernels allows us to apply these fast summation techniques to vortex particle methods, since the field derivatives for the velocity and the vorticity update can be obtained by formal differentiation of the expansion. In the following, we examine algebraic smoothing kernels and their properties, which prove to be very helpful for the analysis of multipole expansions of Section 3.2.

3.1.1 Definition and smoothing properties

In this section, we focus on algebraic kernels like the zeroth or second order algebraic kernel of Section 2.1.2 and extend their structure to a broader class of functions. For general treatment, we define a template and a class containing all functions of this type.

Definition 3.1.1 (Class of algebraic smoothing kernels)
For $\sigma \geq 0$ we define the *class A_σ of algebraic smoothing kernels* as

$$A_\sigma := \left\{ g \in C^\infty(\mathbb{R}) : \; g(\rho) = \sum_{l=0}^{\lambda} \frac{a_l \, \rho^{2l}}{(\rho^2+\sigma^2)^{\lambda+\frac{1}{2}}}, \; a_l \in \mathbb{R}, \lambda \in \mathbb{N}_0 \right\}.$$

We refer to a member of this class as $g^\lambda \in A_\sigma$ if the parameter $\lambda \in \mathbb{N}_0$ is of interest.

A radially symmetric smoothing kernel ζ (see Definition 2.1.1) in algebraic form can then be characterized as a member of A_1, while the corresponding regularized smoothing kernel

$$\zeta_\sigma(x) = \frac{1}{\sigma^3}\,\zeta\left(\frac{|x|}{\sigma}\right) =: \frac{1}{\sigma^3}\,\zeta\left(\frac{\rho}{\sigma}\right)$$

itself is a member of A_σ. The following example places the already introduced zeroth and second order kernel into the context of A_σ.

Example 3.1.2
We choose a fixed $\sigma > 0$.

- For the zeroth order algebraic kernel and its regularized equivalent

$$\zeta^{(0)}(\rho) = \frac{3}{4\pi(\rho^2+1)^{\frac{5}{2}}} \text{ and } \zeta_\sigma^{(0)}(\rho) = \frac{3\sigma^2}{4\pi(\rho^2+\sigma^2)^{\frac{5}{2}}},$$

  we have $\zeta^{(0)} \in A_1$ with $\lambda = 2$, $a_0 = \frac{3}{4\pi}$, $a_1 = a_2 = 0$ and $\zeta_\sigma^{(0)} \in A_\sigma$ with $\lambda = 2$, $a_0 = \frac{3\sigma^2}{4\pi}$, $a_1 = a_2 = 0$.

- For the second order algebraic kernel

$$\zeta^{(2)}(\rho) = \frac{15}{8\pi(\rho^2+1)^{\frac{7}{2}}} \text{ and } \zeta_\sigma^{(2)}(\rho) = \frac{15\sigma^4}{8\pi(\rho^2+\sigma^2)^{\frac{7}{2}}},$$

  we have $\zeta^{(2)} \in A_1$ with $\lambda = 3$, $a_0 = \frac{15}{8\pi}$, $a_1 = a_2 = a_3 = 0$ and $\zeta_\sigma^{(2)} \in A_\sigma$ with $\lambda = 3$, $a_0 = \frac{15\sigma^4}{8\pi}$, $a_1 = a_2 = a_3 = 0$.

We note that for these smoothing kernels many coefficients a_l are equal to zero. In Remark 2.1.2, we have seen that normalization constraint and moment conditions for a radially symmetric smoothing kernel $\zeta^{(r)}$ of order $r \in \mathbb{N}_0$, r even, are

$$\begin{aligned}
I_0 &:= \int_0^\infty \zeta^{(r)}(\rho)\, \rho^2 \mathrm{d}\rho = \frac{1}{4\pi}, \\
I_s &:= \int_0^\infty \zeta^{(r)}(\rho)\, \rho^{2+s} \mathrm{d}\rho = 0 \text{ for } 2 \le s \le r-1, s \text{ even}, \\
I_r &:= \int_0^\infty |\zeta^{(r)}(\rho)\,| \rho^{2+r} \mathrm{d}\rho < \infty.
\end{aligned}$$

We now analyze the existence of these moment integrals for algebraic functions $\zeta^{(r)} \in A_1$.

Theorem 3.1.3 (Existence of moment integrals)
The integrals I_0, I_s, I_r of Remark 2.1.2 with $r = 2k$, $k \in \mathbb{N}_0$, exist for all functions $g^\lambda \in A_1$, $\lambda \in \mathbb{N}$, $\lambda \ge 2$ with

$$a_l = 0 \text{ for all } l = \lambda - 1 - \frac{r}{2}, \ldots, \lambda.$$

Proof. For a fixed $\lambda \in \mathbb{N}_0$ we choose $g^\lambda \in A_1$ with

$$g^\lambda(\rho) = \sum_{l=0}^{\lambda} \frac{a_l \, \rho^{2l}}{(\rho^2+1)^{\lambda+\frac{1}{2}}}.$$

Inserting this into I_0, we obtain

$$I_0 = \int_0^\infty g^\lambda(\rho)\rho^2 \mathrm{d}\rho = \sum_{l=0}^{\lambda} a_l \int_0^\infty \frac{\rho^{2l+2}}{(\rho^2+1)^{\lambda+\frac{1}{2}}} \mathrm{d}\rho.$$

With Theorem C.6 this yields

$$I_0 = \sum_{l=0}^{\lambda} a_l \, \frac{\Gamma(l+\frac{3}{2})\Gamma(\lambda-l-1)}{2\Gamma(\lambda+\frac{1}{2})}.$$

However, this is only true for $l < \lambda - 1$, yielding a first restriction on l.
Similarly, we have

$$I_s = \sum_{l=0}^{\lambda} a_l \, \frac{\Gamma(l+\frac{3}{2})\Gamma(\lambda-l-1-\frac{s}{2})}{2\Gamma(\lambda+\frac{1}{2})},$$
$$I_r \leq \sum_{l=0}^{\lambda} |a_l| \, \frac{\Gamma(l+\frac{3}{2})\Gamma(\lambda-l-1-\frac{r}{2})}{2\Gamma(\lambda+\frac{1}{2})},$$

which is only valid for $l < \lambda - 1 - \frac{s}{2}$ or $l < \lambda - 1 - \frac{r}{2}$, respectively. □

All algebraic functions satisfying this condition are therefore preliminary candidates for smoothing kernels. Although this theorem only makes a statement about the existence of the moment integrals, its proof gives us a direction for deriving explicit algebraic smoothing kernels of arbitrary order $r = 2k$, $k \in \mathbb{N}_0$, improving e.g. the speed of convergence of the method according to Theorem 2.1.8. We demonstrate this approach with the help of the already known second order algebraic kernel $\zeta^{(2)}$ and illustratively derive a new sixth order algebraic kernel $\zeta^{(6)}$.

Example 3.1.4
Since $\zeta^{(2)}$ is of order two, we only have to satisfy $I_0 = \frac{1}{4\pi}$ and $I_2 < \infty$. From Theorem 3.1.3 we know that $a_l = 0$ for $l \in \{\lambda-2, \lambda-1, \lambda\}$, so that $\lambda = 3$ is required. Therefore, we have to determine a_0 from the first moment condition. With

$$I_0 = \sum_{l=0}^{\lambda} a_l \, \frac{\Gamma(l+\frac{3}{2})\Gamma(\lambda-l-1)}{2\Gamma(\lambda+\frac{1}{2})}$$

we have

$$\frac{1}{4\pi} = a_0 \frac{\Gamma(\frac{3}{2})\Gamma(2)}{2\Gamma(\frac{7}{2})}$$

or $a_0 = \frac{15}{8\pi}$. Furthermore, $I_r = \frac{2a_0}{15} < \infty$, so that we have reconstructed

$$\zeta^{(2)}(\rho) = \frac{15}{8\pi(\rho^2+1)^{\frac{7}{2}}}.$$

To construct $\zeta^{(6)}$, we note that Theorem 3.1.3 gives us $a_l = 0$ for $l = \lambda - 4, \ldots, \lambda$, so that $\lambda > 4$ and $a_0 \neq 0$. However, we cannot choose $\lambda = 5$ since we obtain three moment conditions

$$\begin{aligned} I_0 &= \frac{1}{4\pi}, \\ I_2 &= 0, \\ I_4 &= 0 \end{aligned}$$

and $I_6 < \infty$. While the latter is automatically satisfied by Theorem 3.1.3, we require $\lambda = 7$ with $a_0, a_1, a_2 \in \mathbb{R}\backslash\{0\}$ for the three moment conditions, so that we can initially define $\zeta^{(6)}$ using the template

$$\zeta^{(6)} = \frac{a_0 + a_1\rho^2 + a_2\rho^4}{(\rho^2+1)^{\frac{15}{2}}}.$$

The three moment conditions yield the following system of equations

$$\begin{pmatrix} \Gamma(6)\Gamma(3/2) & \Gamma(5)\Gamma(5/2) & \Gamma(4)\Gamma(7/2) \\ \Gamma(5)\Gamma(5/2) & \Gamma(4)\Gamma(7/2) & \Gamma(3)\Gamma(9/2) \\ \Gamma(4)\Gamma(7/2) & \Gamma(3)\Gamma(9/2) & \Gamma(2)\Gamma(11/2) \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} \frac{\Gamma(15/2)}{2\pi} \\ 0 \\ 0 \end{pmatrix}.$$

This leads to

$$a_0 = \frac{11025}{512\pi}, \quad a_1 = -\frac{6615}{128\pi}, \quad a_2 = \frac{945}{64\pi},$$

and therefore

$$\zeta^{(6)}(\rho) = \frac{315}{512\pi} \frac{35 - 84\rho^2 + 24\rho^4}{(\rho^2+1)^{\frac{15}{2}}},$$

which concludes this example.

The disadvantage of the current formulation of the class A_σ is adumbrated in the proof of Theorem 3.1.3: even linear operators applied to members of this class – like the integral evaluation or derivatives – can already involve complicated calculations. An elegant circumvention of this problem is presented in the next section.

3.1.2 A decomposition theorem

Although Theorem 3.1.3 ensures the existence of the moment integrals, for ζ being a suitable smoothing kernel in the sense of Theorem 2.1.8, we still have to verify

$$\zeta \in W^{m,\infty}(\mathbb{R}^3) \cap W^{m,1}(\mathbb{R}^3) \text{ for all } m \in \mathbb{N}.$$

Here, the following result proves very helpful: the proposed decomposition formula simplifies the handling of functions $g \in A_\sigma$ radically.

Theorem 3.1.5 (Decomposition Theorem)
For all $\lambda \in \mathbb{N}_0$, $\rho > 0$ and $g^\lambda \in A_\sigma$ with $\sigma > 0$, we have

$$g^\lambda(\rho) = \sum_{l=0}^{\lambda} \frac{a_{\lambda-l}^{(l)}}{(\rho^2+\sigma^2)^{l+\frac{1}{2}}}$$

with

$$\begin{aligned} a_\nu^{(0)} &:= a_\nu \text{ for all } \nu = 0, \dots, \lambda, \\ a_\nu^{(\mu)} &:= a_\nu^{(\mu-1)} - a_{\lambda-(\mu-1)}^{(\mu-1)} \sigma^{2(\lambda-\mu+1)-2\nu} \binom{\lambda-(\mu-1)}{\nu} \text{ for all } \nu = 0, \dots, \lambda-\mu. \end{aligned}$$

Proof. The idea of this decomposition of $g := g^\lambda$ is the elimination of ρ-terms in the numerator. Let $\lambda \in \mathbb{N}_0$ and $g = g^\lambda \in A_\sigma$. We write for $\rho > 0$

$$\begin{aligned} g(\rho) &= \sum_{l=0}^{\lambda} \frac{a_l \, \rho^{2l}}{(\rho^2+\sigma^2)^{\lambda+\frac{1}{2}}} \\ &= \sum_{l=0}^{\lambda} \frac{a_l \, \rho^{2l}}{(\rho^2+\sigma^2)^{\lambda+\frac{1}{2}}} + \frac{a_\lambda}{(\rho^2+\sigma^2)^{\frac{1}{2}}} - a_\lambda \sum_{l=0}^{\lambda} \binom{\lambda}{l} \frac{\rho^{2l} \, \sigma^{2\lambda-2l}}{(\rho^2+\sigma^2)^{\lambda+\frac{1}{2}}}, \end{aligned}$$

after making use of the binomial theorem for $(\rho^2+\sigma^2)^\lambda$. This expression can be combined into

$$g(\rho) = \frac{a_\lambda}{(\rho^2+\sigma^2)^{\frac{1}{2}}} + \sum_{l=0}^{\lambda} \left(a_l - a_\lambda \sigma^{2\lambda-2l} \binom{\lambda}{l} \right) \frac{\rho^{2l}}{(\rho^2+\sigma^2)^{\lambda+\frac{1}{2}}}.$$

The last term is now of interest: For $l = \lambda$ this formulation allows us to eliminate the $\rho^{2\lambda}$-term, since

$$a_\lambda - a_\lambda \sigma^{2\lambda-2\lambda} \binom{\lambda}{\lambda} = 0$$

and therefore

$$g(\rho) = \frac{a_\lambda}{(\rho^2+\sigma^2)^{\frac{1}{2}}} + \sum_{l=0}^{\lambda-1} \left(a_l - a_\lambda \sigma^{2\lambda-2l} \binom{\lambda}{l} \right) \frac{\rho^{2l}}{(\rho^2+\sigma^2)^{\lambda+\frac{1}{2}}}.$$

To obtain the asserted recursion formula for $a_\nu^{(\mu)}$, we define

$$\begin{aligned} a_l^{(0)} &:= a_l \text{ for all } l = 0 \dots \lambda, \\ a_l^{(1)} &:= a_l^{(0)} - a_\lambda^{(0)} \sigma^{2\lambda-2l} \binom{\lambda}{l} \text{ for all } l = 0 \dots \lambda - 1. \end{aligned}$$

Then we get

$$\begin{aligned} g^{(0)}(\rho) := g(\rho) &= \frac{a_\lambda^{(0)}}{(\rho^2+\sigma^2)^{\frac{1}{2}}} + \sum_{l=0}^{\lambda-1} a_l^{(1)} \frac{\rho^{2l}}{(\rho^2+\sigma^2)^{\lambda+\frac{1}{2}}} \\ &=: \frac{a_\lambda^{(0)}}{(\rho^2+\sigma^2)^{\frac{1}{2}}} + \frac{1}{\rho^2+\sigma^2} \cdot g^{(1)}(\rho). \end{aligned}$$

Applying the same procedure to $g^{(1)}(\rho)$, we obtain

$$\begin{aligned} g^{(1)}(\rho) &= \frac{a_{\lambda-1}^{(1)}}{(\rho^2+\sigma^2)^{\frac{1}{2}}} + \sum_{l=0}^{\lambda-2} \frac{a_l^{(2)} \rho^{2l}}{(\rho^2+\sigma^2)^{\lambda-1+\frac{1}{2}}} \\ &=: \frac{a_{\lambda-1}^{(1)}}{(\rho^2+\sigma^2)^{\frac{1}{2}}} + \frac{1}{\rho^2+\sigma^2} \cdot g^{(2)}(\rho) \end{aligned}$$

for $a_l^{(2)} := a_l^{(1)} - a_{\lambda-1}^{(1)} \sigma^{2\lambda-4-2l} \binom{\lambda-1}{l}$, which eliminates the $\rho^{2\lambda-2}$-term. We can perform this procedure recursively until reaching

$$g^{(\lambda)}(\rho) = \frac{a_0^{(\lambda)}}{(\rho^2+\sigma^2)^{\frac{1}{2}}},$$

which finally yields

$$g(\rho) = \sum_{l=0}^{\lambda} \frac{a_{\lambda-l}^{(l)}}{(\rho^2+\sigma^2)^{l+\frac{1}{2}}}.$$

□

This result will also ease the analysis of multipole expansions and is therefore considered the main statement for the class A_σ of algebraic functions. To demonstrate the decomposition procedure we return to our previous Example 3.1.4 and derive their corresponding decompositions.

Example 3.1.6
While the zeroth and second order kernels are already decomposed by definition, we have for the sixth order kernel

$$\zeta^{(6)}(\rho) = \frac{315}{512\pi}\,\frac{35-84\rho^2+24\rho^4}{(\rho^2+1)^{\frac{15}{2}}} =: \frac{315}{512\pi}\,\frac{a_0+a_1\rho^2+a_2\rho^4}{(\rho^2+1)^{\frac{15}{2}}}.$$

Since $a_3 = \ldots = a_7 = 0$ we extract $a_2 \neq 0$ and obtain

$$\begin{aligned}\zeta^{(6)}(\rho) &= \frac{315}{512\pi}\,\frac{a_0+a_1\rho^2+a_2\rho^4}{(\rho^2+1)^{\frac{15}{2}}} = \frac{a_2}{(\rho^2+1)^{\frac{11}{2}}} + \sum_{l=0}^{2}\left(a_l - a_2\binom{2}{l}\right)\frac{\rho^{2l}}{(\rho^2+1)^{\frac{15}{2}}} \\ &= \frac{315}{512\pi}\left[\frac{24}{(\rho^2+1)^{\frac{11}{2}}} + \sum_{l=0}^{1}\left(a_l - 24\binom{2}{l}\right)\frac{\rho^{2l}}{(\rho^2+1)^{\frac{15}{2}}}\right] \\ &= \frac{315}{512\pi}\left[\frac{24}{(\rho^2+1)^{\frac{11}{2}}} + \frac{11-132\rho^2}{(\rho^2+1)^{\frac{15}{2}}}\right] =: \frac{315}{512\pi}\left[\frac{24}{(\rho^2+1)^{\frac{11}{2}}} + \frac{a_0^{(1)}+a_1^{(1)}\rho^2}{(\rho^2+1)^{\frac{15}{2}}}\right].\end{aligned}$$

Extracting now $a_1^{(1)} = -132$ finally yields

$$\zeta^{(6)}(\rho) = \frac{315}{512\pi}\left[\frac{24}{(\rho^2+1)^{\frac{11}{2}}} - \frac{132}{(\rho^2+1)^{\frac{13}{2}}} + \frac{143}{(\rho^2+1)^{\frac{15}{2}}}\right],$$

which concludes this example.

Next, we abstract the different components of the decomposition method into a new class D_σ.

Definition 3.1.7 (Decomposition Class)
We define the *decomposition class* D_σ as

$$D_\sigma := \left\{ g \in C^\infty(\mathbb{R}) :\ g(\rho) = \frac{1}{(\rho^2+\sigma^2)^\tau},\ \tau > 0 \right\}.$$

Remark 3.1.8
Using the Decomposition Theorem, we can interpret the class D_σ as extending basis of the class A_σ, $\sigma \geq 0$. Conversely, the proof of the Decomposition Theorem yields an extension $\bar{A}_\sigma$ of A_σ. Defining

$$\bar{A}_\sigma := \left\{ g \in C^\infty(\mathbb{R}) :\ g(\rho) = \sum_{l=0}^{\lambda} \frac{a_l\,\rho^{2l}}{(\rho^2+\sigma^2)^{\lambda+s}},\ a_l \in \mathbb{R}, \lambda \in \mathbb{N}_0, s > 0 \right\},$$

we can show that D_σ creates $\bar{A}_\sigma$ if we use linear combination of D_σ-functions. Therefore, the class $\bar{A}_\sigma$ can be seen as the linear hull of D_σ. This is not true for A_σ, since $A_\sigma \subsetneq \bar{A}_\sigma$. For instance, we have

$$\frac{1}{(\rho^2+\sigma^2)^\tau} \notin A_\sigma \text{ for } \tau \neq \lambda + \frac{1}{2}, \lambda \in \mathbb{N}_0.$$

Furthermore, we note that

$$\frac{1}{\rho^n} \in D_0 \text{ for all } n \in \mathbb{N},$$

so that a statement for members of D_σ automatically covers common ρ^{-n}-kernels as well.

Following the Decomposition Theorem, a linear combination of functions $g_\sigma^\tau \in D_\sigma$ for different $\tau = l + \frac{1}{2}$, $l \in \mathbb{N}_0$, resembles each function $g \in A_\sigma$, so that a statement for members of D_σ can often be transferred to members of A_σ and $\bar{A}_\sigma$. With this Theorem 3.1.5 at hand we are now able to prove that all functions $\zeta^{(r)} \in A_1$ satisfying the normalization constraint and moment conditions are potential smoothing kernels in the sense of Theorem 2.1.8.

Theorem 3.1.9
If the moment integrals I_0, I_s and I_r of Remark 2.1.2 exist for a function $g^\lambda \in A_1$, $\lambda \in \mathbb{N}_0$, $\lambda \geq 2$, and $r = 2k$ with $k \in \mathbb{N}_0$, then we have

$$g^\lambda \in W^{m,\infty}(\mathbb{R}^3) \cap W^{m,1}(\mathbb{R}^3) \text{ for all } m \in \mathbb{N}.$$

Proof. Let $g := g^\lambda \in A_1$ for $\lambda \in \mathbb{N}_0$, $\lambda \geq 2$. Since $g \in C^\infty(\mathbb{R}^3)$ we have to verify for the β-th derivative $D^\beta g$ of g

$$D^\beta g \in L^1(\mathbb{R}^3) \cap L^\infty(\mathbb{R}^3) \text{ for all } |\beta| \leq m \in \mathbb{N},$$

according to Theorem B.4 and Definition B.5.
With the Decomposition Theorem, we obtain

$$g(\rho) = \sum_{l=0}^{\lambda} \frac{b_l}{(\rho^2+1)^{l+\frac{1}{2}}} =: \sum_{l=0}^{\lambda} b_l \, g_l(\rho)$$

with b_l as stated in 3.1.5. Next, for multi-indices β, α and $\rho = |x|$ we have

$$D^\beta g_l(x) = \sum_{|\alpha| \leq |\beta|} b_{\alpha,l} \frac{x^\alpha}{(|x|^2+1)^{l+\frac{1}{2}+|\alpha|}}$$

for appropriate coefficients $b_{\alpha,l}$. This yields

$$\left|D^\beta g(x)\right| \leq \sum_{l=0}^{\lambda} \sum_{|\alpha|\leq|\beta|} |b_{\alpha,l}| \, \frac{|x|^{|\alpha|}}{(|x|^2+1)^{l+\frac{1}{2}+|\alpha|}}.$$

We consider

$$\int_{\mathbb{R}^3} \left|D^\beta g(x)\right| \mathrm{d}x \leq 4\pi \sum_{l=0}^{\lambda} \sum_{|\alpha|\leq|\beta|} |b_{\alpha,l}| \int_0^\infty \frac{\rho^{|\alpha|+2}}{(\rho^2+1)^{l+\frac{1}{2}+|\alpha|}} \mathrm{d}\rho,$$

which exists for $l \geq 2$ according to Theorem C.6, where $a-1 := |\alpha|+2$ and $b := l+\frac{1}{2}+|\alpha|$, so that $2b > a$ only if $l \geq 2$.
It follows

$$D^\beta g \in L^1(\mathbb{R}^3) \text{ for all } |\beta| \leq m \in \mathbb{N},$$

so that $g = g^\lambda \in W^{m,1}(\mathbb{R}^3)$ for all $m \in \mathbb{N}$ if $g_l(\rho) \equiv 0$ for $l = 0, 1$. This leads to

$$\begin{aligned} 0 = b_0 &= a_\lambda^{(0)} = a_\lambda, \\ 0 = b_1 &= a_{\lambda-1}^{(0)} - a_\lambda^{(0)} \, \lambda = a_{\lambda-1} - a_\lambda \, \lambda, \end{aligned}$$

so that we require $a_\lambda = a_{\lambda-1} = 0$. This is already true by Theorem 3.1.3, since we assumed the existence of I_0, I_s and I_r.
Moreover, we have

$$\sup_{x\in\mathbb{R}^3} \left|D^\beta g(x)\right| \leq \sup_{x\in\mathbb{R}^3} \sum_{l=0}^{\lambda} \sum_{|\alpha|\leq|\beta|} |b_{\alpha,l}| \, \frac{|x|^{|\alpha|}}{(|x|^2+1)^{l+\frac{1}{2}+|\alpha|}}$$

and since $|x|^{|\alpha|} \leq (|x|^2+1)^{|\alpha|}$, we obtain

$$\sup_{x\in\mathbb{R}^3} \left|D^\beta g(x)\right| \leq \sup_{x\in\mathbb{R}^3} \sum_{l=0}^{\lambda} \sum_{|\alpha|\leq|\beta|} |b_{\alpha,l}| \, \frac{1}{(|x|^2+1)^{l+\frac{1}{2}}} = \sum_{l=0}^{\lambda} \sum_{|\alpha|\leq|\beta|} |b_{\alpha,l}| < \infty.$$

This implies

$$D^\beta g \in L^\infty(\mathbb{R}^3) \text{ for all } |\beta| \leq m \in \mathbb{N}$$

and therefore $g = g^\lambda \in W^{m,\infty}(\mathbb{R}^3)$ for all $m \in \mathbb{N}$. □

Remark 3.1.10
This result motivates the definition of the class S_σ containing all suitable algebraic smoothing kernels. For $\lambda \in \mathbb{N}$, $\lambda \geq 2$, $\sigma \geq 0$ and $r = 2k$, $k \in \mathbb{N}_0$, we define the *class S_σ^r of suitable algebraic smoothing kernels* as

$$S_\sigma^r := \left\{ g \in C^\infty(\mathbb{R}) : \; g(\rho) = \sum_{l=0}^{\lambda-1-\frac{r}{2}} \frac{s_l \, \rho^{2l}}{(\rho^2+\sigma^2)^{\lambda+\frac{1}{2}}} \; s_l \in \mathbb{R}, \lambda \in \mathbb{N}_0 \right\}.$$

With Theorem 3.1.9 we can therefore conclude that all functions $\zeta^{(r)} \in S_\sigma^r$ with

$$\begin{aligned} I_0 &= \int_0^\infty \zeta^{(r)}(\rho) \, \rho^2 \mathrm{d}\rho = \frac{1}{4\pi}, \\ I_s &= \int_0^\infty \zeta^{(r)}(\rho) \, \rho^{2+s} \mathrm{d}\rho = 0 \quad \text{for } 2 \leq s \leq r-1, s \text{ even}, \\ I_r &= \int_0^\infty |\zeta^{(r)}(\rho)| \rho^{2+r} \mathrm{d}\rho < \infty \end{aligned}$$

are smoothing kernels as needed for a convergent vortex particle method. For instance, we have

$$\zeta^{(0)} \in S_1^0, \;\; \zeta^{(2)} \in S_1^2 \;\text{ and }\; \zeta^{(6)} \in S_1^6.$$

3.1.3 From smoothing kernels to stream functions

Next, we can construct the kernel G for the stream function ψ. As we have seen in Theorem 2.1.3, discretized and regularized vorticity and stream function are coupled by

$$\Delta \tilde{\psi}_\sigma = -\tilde{\omega}_\sigma,$$

while ζ and G are connected via

$$-\zeta(\rho) = \frac{1}{\rho} \frac{\mathrm{d}^2}{\mathrm{d}\rho^2} \left(\rho \, G(\rho) \right).$$

The next theorem places G into the context of the class A_1. Naturally, the same result applies to the classes $\bar{A}_1$ of Remark 3.1.8 and S_1^r of Remark 3.1.10 as well.

Theorem 3.1.11
For $\zeta^\lambda \in A_1$ with $\lambda \in \mathbb{N}$, $\lambda \geq 2$, we have $G^{\lambda'} \in A_1$ with $\lambda' = \lambda - 2$.

Proof. We define $\lambda' := \lambda - 2$ for $\lambda \geq 2$ and

$$G^{\lambda'}(\rho) := G(\rho) := \sum_{l=0}^{\lambda'} \frac{b_l \, \rho^{2l}}{(\rho^2+1)^{\lambda'+\frac{1}{2}}}.$$

Then

$$\frac{\mathrm{d}^2}{\mathrm{d}\rho^2}\left(\rho\, G(\rho)\right) = \sum_{l=0}^{\lambda'} b_l \left(\rho^{2l-1} \sum_{i=0}^{2} c_i^l \rho^{2i}\right) \frac{1}{(\rho^2+1)^{\lambda'+2+\frac{1}{2}}}$$

for adequate choices of c_i^l, but with $c_0^0 = c_2^{\lambda'} = 0$, so that the powers of ρ are in the range of 1 to $2\lambda'+1$. A straightforward index shift of the outer sum leads to

$$\begin{aligned} -\zeta^\lambda(\rho) =& \frac{1}{\rho}\frac{\mathrm{d}^2}{\mathrm{d}\rho^2}\left(\rho\, G(\rho)\right) = \frac{1}{\rho}\sum_{l=0}^{\lambda'} \frac{\tilde{b}_l\, \rho^{2l+1}}{(\rho^2+1)^{\lambda'+2+\frac{1}{2}}} \\ =:& -\sum_{l=0}^{\lambda} \frac{a_l\, \rho^{2l}}{(\rho^2+1)^{\lambda+\frac{1}{2}}}, \end{aligned}$$

with appropriate coefficients $\tilde{b}_l$ and $a_\lambda = a_{\lambda-1} = 0$ by the definition of λ'.
Performing the formal inverse operations of this procedure with suitable choices of the integration constants concludes the proof of this theorem. □

Remark 3.1.12
Similarly, we can show that for $\sigma > 0$ and $\zeta_\sigma^\lambda \in A_\sigma$ we again obtain $G_\sigma^{\lambda'} \in A_\sigma$ with $\lambda' = \lambda - 2$ and

$$G_\sigma^{\lambda'}(\rho) = \frac{1}{\sigma}\, G^{\lambda'}\left(\frac{\rho}{\sigma}\right).$$

We continue the previous Examples 3.1.4 and 3.1.6 of second and sixth order smoothing kernels $\zeta^{(2)}, \zeta^{(6)}$ and state the corresponding stream function kernels $G_\sigma^{(2)}$ and $G_\sigma^{(6)}$. We use the formal inversion as stated in the proof of Theorem 3.1.11.

Example 3.1.13

a) For $\zeta^{(2)}$ we obtain

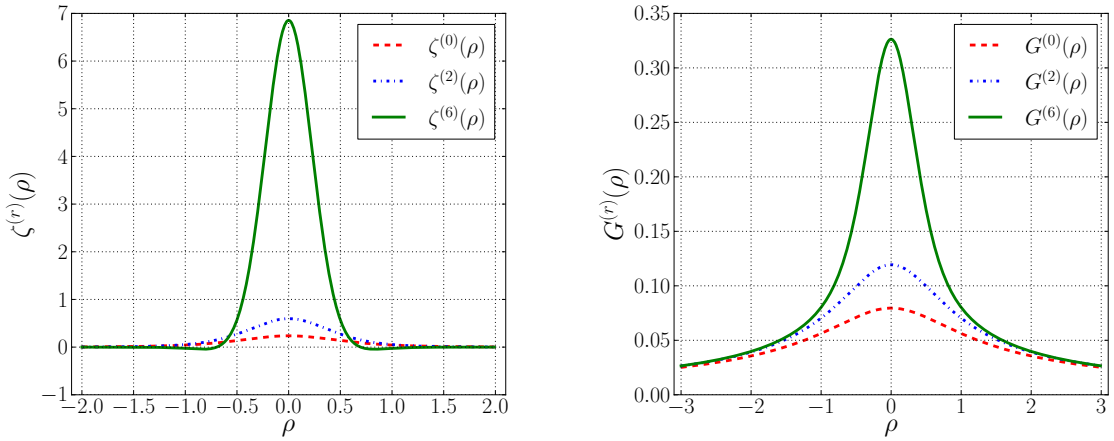
$$G^{(2)}(\rho) = -\frac{1}{\rho}\iint \rho\, \zeta^{(2)}(\rho)\, \mathrm{d}\rho\, \mathrm{d}\rho = \frac{1}{4\pi}\frac{\rho^2+\frac{3}{2}}{(\rho^2+1)^{\frac{3}{2}}}$$

and therefore

$$G_\sigma^{(2)}(\rho) = \frac{1}{\sigma}\, G^{(2)}\left(\frac{\rho}{\sigma}\right) = \frac{1}{8\pi}\frac{2\rho^2+3\sigma^2}{(\rho^2+\sigma^2)^{\frac{3}{2}}}.$$

b) For $\zeta^{(6)}$ we obtain similarily

$$G_\sigma^{(6)}(\rho) = \frac{1}{512\pi}\frac{128\rho^{10}+704\sigma^2\rho^8+1584\sigma^4\rho^6+1848\sigma^6\rho^4+1050\sigma^8\rho^2+525\sigma^{10}}{(\rho^2+\sigma^2)^{\frac{11}{2}}}.$$



(a) Zeroth, second and sixth order algebraic smoothing kernels

(b) Zeroth, second and sixth order algebraic stream function kernels

Figure 3.2: Visualization of three different algebraic smoothing kernels ζ and the corresponding stream function kernels G. While the zeroth and second order functions are strictly positive, the sixth order function has negative parts. This behavior does not recur within the corresponding G.

c) Furthermore, the decompositions for this functions $G_\sigma^{(2)}$ and $G_\sigma^{(6)}$ in the sense of Theorem 3.1.5 are given by

$$G_\sigma^{(2)}(\rho) = \frac{1}{8\pi}\left(\frac{2}{(\rho^2+\sigma^2)^{\frac{1}{2}}} + \frac{\sigma^2}{(\rho^2+\sigma^2)^{\frac{3}{2}}}\right),$$

$$G_\sigma^{(6)}(\rho) = \frac{1}{512\pi}\left(\frac{128}{(\rho^2+\sigma^2)^{\frac{1}{2}}} + \frac{64\sigma^2}{(\rho^2+\sigma^2)^{\frac{3}{2}}} + \frac{48\sigma^4}{(\rho^2+\sigma^2)^{\frac{5}{2}}} + \frac{40\sigma^6}{(\rho^2+\sigma^2)^{\frac{7}{2}}} - \frac{70\sigma^8}{(\rho^2+\sigma^2)^{\frac{9}{2}}} + \frac{315\sigma^{10}}{(\rho^2+\sigma^2)^{\frac{11}{2}}}\right).$$

Figure 3.2 shows the functions $\zeta^{(0)}$, $\zeta^{(2)}$ and $\zeta^{(6)}$ together with their derived stream function kernels $G^{(0)}$, $G^{(2)}$ and $G^{(6)}$. It is important to note that smoothing kernels $\zeta^{(r)}$ of order $r > 2$ inherently take positive and negative values, since $I_s = 0$ has to be satisfied.

In this section, we have seen that smoothing kernels ζ, regularized kernels ζ_σ and regularized stream function kernels G_σ belong to the class A_σ or $S_\sigma^r \subset A_\sigma$, respectively. Therefore, our goal is to formulate a theory of multipole expansions at least for members of S_σ^r. To this end, the introduction of the component class D_σ will prove of great help, since multipole expansions rely on explicit treatment of derivatives of the underlying kernels. Since

all upcoming operators and procedures are linear, the results can be directly transferred to the classes $S_\sigma^r \subset A_\sigma$, A_σ and even $\bar{A}_\sigma$. Before we deduce the expansion theory for functions in D_σ, we introduce the mathematical background of general multipole expansions and their error terms. The definitions and theorems of Appendix C are required for the focus on rotationally invariant kernels, which will directly lead us to a theory for D_σ.

3.2 Multipole expansions in the context of algebraic kernels

To embed the idea of multipole expansions into vortex particle methods, we focus on the Poisson equation in the continuous, regularized form

$$\Delta\psi_\sigma = -\omega_\sigma$$

with $\omega_\sigma = \omega * \zeta_\sigma$. The Green's function approach then gives us

$$\psi_\sigma(x) = G_\sigma(x) * \omega(x) = \int G_\sigma(x-y)\,\omega(y)\mathrm{d}y,$$

where we have omitted the time dependency for brevity.

Now, to compute the effect on a particular point x of a distribution of particles and strengths inside a discretized volume $\mathcal{V}$, we choose a particular point $\hat{x} \in \mathcal{V}$, which for convenience is usually taken as the 'center-of-charge', and write

$$\tilde{\psi}_\sigma(x) = \sum_{y\in\mathcal{V}} G_\sigma(x-y)\cdot\omega(y)\mathrm{vol}(y) = \sum_{y\in\mathcal{V}} G_\sigma((x-\hat{x})-(y-\hat{x}))\cdot\alpha_y$$

for $x \neq \hat{x}$ and strength vector $\alpha_y = \omega(y)\mathrm{vol}(y)$.
A Taylor series expansion in three dimensions of $G_\sigma(x-y)$ yields

$$\begin{aligned} G_\sigma(x-y) &= G_\sigma((x-\hat{x})-(y-\hat{x})) \\ &= \sum_{j=0}^{\infty}\left\{\frac{(-1)^j}{j!}\left(\sum_{k=1}^{3}(y_k-\hat{x}_k)\frac{\partial}{\partial z_k}\right)^j G_\sigma(z)\right\}_{z=x-\hat{x}} \end{aligned}$$

or using multi-index notation (see Definition A.3 for details)

$$G_\sigma((x-\hat{x})-(y-\hat{x})) = \sum_{|\beta|\geq 0}\frac{(-1)^{|\beta|}}{\beta!}(y-\hat{x})^\beta\cdot D^\beta G_\sigma(z)\big|_{z=x-\hat{x}}$$

for $y \in \mathbb{R}^3$. We can insert this expression into $\tilde{\psi}_\sigma(x)$ and obtain the series expansion

$$\tilde{\psi}_\sigma(x) = \sum_{|\beta| \geq 0} \phi_{(\beta)}(x)$$

with $\phi_{(\beta)}(x)$ as defined below. Since an exact evaluation of $\tilde{\psi}_\sigma(x)$ would imply an infinite summation, any practical implementation requires a truncation of this sum, leading to some finite error.

Definition 3.2.1 (Fundamentals of multipole expansions)
We define the analytical finite *multipole expansion* for $\tilde{\psi}_\sigma$ of order $p \in \mathbb{N}_0$ with *error term* $\varepsilon_{(p)}$ as

$$\tilde{\psi}_\sigma(x) = \sum_{|\beta|=0}^{p} \phi_{(\beta)}(x) + \varepsilon_{(p+1)}(x)$$

with multi-index β and

$$\begin{aligned}
\phi_{(\beta)}(x) &:= \frac{(-1)^{|\beta|}}{|\beta|!} M^{(\beta)} \cdot D^\beta G_\sigma(z)\big|_{z=x-\hat{x}}, \\
M^{(\beta)} &:= \sum_{y \in \mathcal{V}} (y - \hat{x})^\beta \cdot \alpha_y, \\
\varepsilon_{(p+1)}(x) &:= \sum_{|\beta|=p+1} \sum_{y \in \mathcal{V}} \kappa_{(\beta)}(x, y) \cdot \alpha_y.
\end{aligned}$$

The error kernel $\kappa_{(\beta)}$ in integral form for $|\beta| = p + 1$ is given by

$$\kappa_{(\beta)}(x, y) = \frac{(-1)^{p+1}}{p!} \cdot (y - \hat{x})^\beta \cdot \int_0^1 (1 - t)^p \cdot D^\beta G_\sigma(z(t))\big|_{z(t)=x-\hat{x}-t(y-\hat{x})} \mathrm{d}t.$$

The factors $M^{(\beta)} \in \mathbb{R}^3$ are called *multipole moments*.

This is the definition of the multipole expansion for arbitrary functions G_σ. Although being a formal Taylor expansion, the coupling to physical multipole moments suggests the name 'multipole expansion', which we will use in the following sections. Writing $\phi_{(\beta)}(x)$ as a product of multipole moments $M^{(\beta)}$ and derivatives of G_σ has one important advantage: since the multipole moments no longer depend on the particular point x but only on members y of the cluster $\mathcal{V}$, these can be precomputed, as we will see in Section 4.1.2. Equally, the derivatives of G_σ are independent of the actual particles of $\mathcal{V}$ but only depend on their common center-of-charge $\hat{x}$. Their implementation can be decoupled from the computation of multipole moments. This decomposition of $\phi_{(\beta)}(x)$ yields the fundamental principle of a multipole-based tree algorithm.

Next, we apply this concept firstly to rotationally invariant kernels and finally to kernels

$$G_\sigma^\tau \in D_\sigma,$$

yielding in the context of the Decomposition Theorem a comprehensive multipole expansion theory for the class A_σ.

3.2.1 Notations for rotationally invariant kernels

In this section, we briefly review the ideas and notations of John K. Salmon [90]. The upcoming notations and abbreviations will be used frequently for the error analysis. We combine them into the following central definition.

Definition 3.2.2
We use the following notations and abbreviations for rotationally invariant kernels G_σ. For a cluster $\mathcal{V}$ with particles y we denote the center-of-charge by $\hat{x}$ and the distance between y and $\hat{x}$ by $c := |y - \hat{x}|$. The maximum extension of $\mathcal{V}$ is given by $b := \sup\{|y - \hat{x}| : y \in \mathcal{V}\}$. For the current particle x the distance to $\hat{x}$ is denoted by $d := |x - \hat{x}|$. Furthermore, we use

$$f\left(\frac{\rho^2}{2}\right) := g(\rho) := G_\sigma(|x|) = G_\sigma(x)$$

$$g^{(j)}(\rho) := \frac{\rho^{2j}}{g(\rho)} \left(\frac{\mathrm{d}^j}{\mathrm{d}s^j} f(s)\right)\Bigg|_{\frac{\rho^2}{2}} \quad \text{for } j \in \mathbb{N}_0$$

$$e := \frac{x - \hat{x}}{d}$$

$$d_t := |x - \hat{x} - t(y - \hat{x})|$$

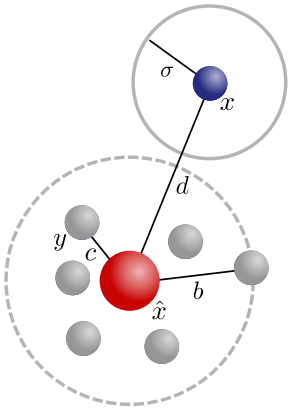
$$e_t := \frac{x - \hat{x} - t(y - \hat{x})}{d_t}$$

$$\mu_t := \frac{(y - \hat{x})e_t}{c}$$

$$\mathcal{Q}_j(t) := \sum_{m=0}^{\lfloor \frac{j}{2} \rfloor} \frac{(-1)^j}{2^m (j-2m)! m!} \cdot g(d_t) g^{(j-m)}(d_t) \cdot \mu_t^{j-2m}$$

$$R_t := \sqrt{(d_t^2 + \sigma^2)}$$

$$v_t := \mu_t \frac{d_t}{R_t}.$$



These notations are stated in order of their appearance.

Following Salmon's notation [90], we can simplify the derivatives D^β of G_σ to

$$\begin{aligned}\frac{\partial^{i_1}}{\partial z_{i_1}} G_\sigma(z)\big|_{z=x-\hat{x}} &= f'\left(\frac{|x-\hat{x}|^2}{2}\right)(x-\hat{x})_{i_1} \\ \frac{\partial^{i_1+i_2}}{\partial z_{i_1}\partial z_{i_2}} G_\sigma(z)\big|_{z=x-\hat{x}} &= f''\left(\frac{|x-\hat{x}|^2}{2}\right)(x-\hat{x})_{i_1}(x-\hat{x})_{i_2} + f'\left(\frac{|x-\hat{x}|^2}{2}\right)\delta_{i_1,i_2} \\ &\vdots\end{aligned}$$

with the Kronecker-symbol δ_{i_1,i_2}. For $j := |\beta|$ we obtain

$$D^\beta G_\sigma(z)\big|_{z=x-\hat{x}} = \frac{g(d)}{d^j} \sum_{m=0}^{\lfloor \frac{j}{2} \rfloor} g^{(j-m)}(d) \cdot T,$$

where T represents the sum over all permutations of m Kronecker-$\delta_{i_k,i_{k+1}}$ and $j-2m$ normalized distances e_{i_k}, so that each

$$T = (\delta_{i_1,i_2} \dots \delta_{i_{2m-1},i_{2m}}\, e_{i_{2m+1}} \dots e_{i_j} \; + \dots + \; \delta_{i_{j-2m+1},i_{j-2m+2}} \dots \delta_{i_{j-1},i_j}\, e_{i_1} \dots e_{i_{j-2m}})$$

consists of $j!/(2^m(j-2m)!m!)$ permutations. To apply this expression to the multipole moments in Definition 3.2.1 we finally define

$$\begin{aligned} M_{(l)}^{(\beta)} &= \delta_{i_{l+1},i_{l+2}} \dots \delta_{i_{j-1},i_j} \cdot M^{(\beta)} \text{ and} \\ \langle M_{(l)}^{(\beta)} | e^{(k)} \rangle &= M_{(l)}^{(\beta)} \cdot e_{i_{l-k+1}} \dots e_{i_l}. \end{aligned}$$

Using this definition, $\phi_{(\beta)}$ of Definition 3.2.1 can be rewritten as

$$\phi_{(\beta)} = \frac{g(d)}{d^j} \sum_{m=0}^{\lfloor \frac{j}{2} \rfloor} \frac{(-1)^j}{2^m (j-2m)! m!} \cdot \langle M_{(j-2m)}^{(\beta)} | e^{(j-2m)} \rangle \cdot g^{(j-m)}(d).$$

We apply the same procedure to the error term $\varepsilon_{(p+1)}$ and obtain

$$\varepsilon_{(p+1)}(x) = \sum_{|\beta|=p+1} \sum_{y \in \mathcal{V}} \kappa_{(\beta)}(x,y) \cdot \alpha_y =: \sum_{y \in \mathcal{V}} \kappa_{(p+1)}(x,y) \cdot \alpha_y$$

with $\kappa_{(p+1)}(x,y) =: \kappa_{(p+1)}$ and

$$\begin{aligned} \kappa_{(p+1)} &= \int_0^1 (1-t)^p \left(\frac{c}{d_t}\right)^{p+1} \sum_{m=0}^{\lfloor \frac{p+1}{2} \rfloor} \frac{(-1)^{p+1}(p+1)}{2^m (p+1-2m)! m!} \cdot g(d_t) g^{(p+1-m)}(d_t) \cdot \mu_t^{p+1-2m} \mathrm{d}t \\ &= (p+1) \int_0^1 (1-t)^p \left(\frac{c}{d_t}\right)^{p+1} \cdot \mathcal{Q}_{p+1}(t) \mathrm{d}t \end{aligned}$$

with Definition 3.2.2. For further details on this deduction, we refer to [90].

We have now reviewed the most important notations of multipole expansions for rotationally invariant kernels. In order to apply this approach to a given problem, we have to specify the kernels G_σ, since both expansion and its remainder need the explicit evaluation of the derivatives of G_σ and therefore $g^{(j-m)}(\rho)$. For regularized vortex particle methods, we have already defined the kernels of interest using the class A_σ and its component class D_σ. In the next section, we will fully exploit the structure of these rotationally invariant kernels and deduce an appropriate and comprehensive multipole expansion theory.

3.2.2 Multipole expansions for decomposed smoothing kernels

We recall the definition of the component class D_σ

$$D_\sigma := \left\{ g \in C^\infty(\mathbb{R}) : \; g(\rho) = \frac{1}{(\rho^2+\sigma^2)^\tau}, \; \tau > 0 \right\}.$$

Since all members of this class are rotationally invariant, the notations from the past section are directly applicable for every $G_\sigma^\tau \in D_\sigma$. Of course, this is also true for every $G_\sigma \in A_\sigma$, but stating an explicit expression for the derivatives of G_σ or f, respectively, and deducing the error terms is not straightforward. Here, the Decomposition Theorem is of great help: a linear combination of these functions resembles every $G_\sigma \in A_\sigma$, so that the generalization of the upcoming theory to A_σ and even $\bar{A}_\sigma$ is straightforward.

Now, our primary goal is to bound the error kernel $\kappa_{(p+1)}$ in terms of the multipole expansion order p. This requires an in-depth estimation of $\mathcal{Q}_{p+1}(t)$. Through the functions $g(d_t)$ and $g^{(p+1-m)}(d_t)$, an explicit kernel $G_\sigma^\tau \in D_\sigma$ and its derivatives influences $\mathcal{Q}_{p+1}(t)$ substantially. Therefore, the first step is to find an explicit statement for the derivatives of $f^\tau(s)$ with

$$f^\tau\left(\frac{\rho^2}{2}\right) = G_\sigma^\tau(\rho)$$

for $G_\sigma^\tau \in D_\sigma$. To this end, the following theorem expresses the derivatives in terms of the Pochhammer symbol $(\tau)_k$ of Definition C.1.

Theorem 3.2.3
For all $\tau > 0$, $s > 0$, $\sigma \geq 0$ and $k \in \mathbb{N}_0$, the k-th derivative of f^τ can be written as

$$\frac{\mathrm{d}^k}{\mathrm{d}s^k} f^\tau(s) = (-1)^k \frac{2^k \; (\tau)_k}{(2s+\sigma^2)^{\tau+k}}.$$

Proof. We prove this result by induction over $k \in \mathbb{N}_0$.
Let $\tau > 0$, $s > 0$, $\sigma \geq 0$, then we obtain for $k = 0$

$$\frac{\mathrm{d}^0}{\mathrm{d}s^0} f^\tau(s) = (-1)^0 \frac{2^0 \ (\tau)_0}{(2s+\sigma^2)^\tau} = \frac{1}{(2s+\sigma^2)^\tau} = f(s),$$

being the basis for our induction.
Assuming that for a given $k \in \mathbb{N}_0$ the expression for the derivatives is correct, we can write

$$\frac{\mathrm{d}^{k+1}}{\mathrm{d}s^{k+1}} f^\tau(s) = \frac{\mathrm{d}}{\mathrm{d}s} \left(\frac{\mathrm{d}^k}{\mathrm{d}s^k} f^\tau(s) \right)$$

and by using the induction hypothesis

$$\begin{aligned}
\frac{\mathrm{d}^{k+1}}{\mathrm{d}s^{k+1}} f^\tau(s) &= \frac{\mathrm{d}}{\mathrm{d}s} \left((-1)^k \frac{2^k \ (\tau)_k}{(2s+\sigma^2)^{\tau+k}} \right) \\
&= (-1)^k \, 2^k \ (\tau)_k \, \frac{\mathrm{d}}{\mathrm{d}s} \left(\frac{1}{(2s+\sigma^2)^{\tau+k}} \right) \\
&= \frac{(-1)^{k+1} \, 2^{k+1}}{(2s+\sigma^2)^{\tau+k+1}} \cdot (\tau)_k \cdot (\tau+k).
\end{aligned}$$

From Theorem C.2 we have

$$(\tau)_k \cdot (\tau+k) = (\tau)_{k+1},$$

so that

$$\frac{\mathrm{d}^{k+1}}{\mathrm{d}s^{k+1}} f^\tau(s) = (-1)^{k+1} \frac{2^{k+1} \ (\tau)_{k+1}}{(2s+\sigma^2)^{\tau+k+1}},$$

which proves the statement for $k+1$. □

With this result we are able to rewrite and bound $\mathcal{Q}_j^\tau(t)$ of Definition 3.2.2 – now for the function G_σ^τ – in terms of Gegenbauer polynomials $\mathcal{C}_j^\tau$, as they are defined in Definition C.3. The terms R_t and v_t are taken from Definition 3.2.2 as well.

Theorem 3.2.4
For all $\tau > 0$, $j \in \mathbb{N}_0$ and $t \in (0,1)$, we have

$$\mathcal{Q}_j^\tau(t) = \frac{1}{R_t^{2\tau}} \cdot \left(\frac{d_t}{R_t} \right)^j \cdot \mathcal{C}_j^\tau(v_t)$$

and

$$|\mathcal{Q}_j^\tau(t)| \leq \frac{1}{R_t^{2\tau}} \cdot \left(\frac{d_t}{R_t} \right)^j \cdot \frac{(2\tau)_j}{j!}.$$

Proof. We start from Definition 3.2.2 of $\mathcal{Q}_j(t)$ or $\mathcal{Q}_j^\tau(t)$, respectively, which gives us

$$\mathcal{Q}_j^\tau(t) = \sum_{m=0}^{\lfloor \frac{j}{2} \rfloor} \frac{(-1)^j}{2^m (j-2m)! m!} \cdot g^\tau(d_t) g^{\tau,(j-m)}(d_t) \cdot \mu_t^{j-2m}$$

with

$$g^\tau(d_t) g^{\tau,(j-m)}(d_t) = f^\tau\left(\frac{d_t^2}{2}\right) \cdot \frac{d_t^{2j-2m}}{g^\tau(d_t)} \frac{\mathrm{d}^{j-m}}{\mathrm{d}s^{j-m}} f^\tau(s)\Big|_{\frac{d_t^2}{2}} = d_t^{2j-2m} \frac{\mathrm{d}^{j-m}}{\mathrm{d}s^{j-m}} f^\tau(s)\Big|_{\frac{d_t^2}{2}}.$$

Theorem 3.2.3 yields

$$g^\tau(d_t) g^{\tau,(j-m)}(d_t) = d_t^{2j-2m} \cdot (-1)^{j-m} \cdot \frac{2^{j-m} \, (\tau)_{j-m}}{(d_t^2 + \sigma^2)^{\tau + j - m}},$$

which in terms of $R_t = \sqrt{(d_t^2 + \sigma^2)}$ can be written as

$$g^\tau(d_t) g^{\tau,(j-m)}(d_t) = (-1)^{j-m} \cdot (\tau)_{j-m} \cdot 2^m \cdot 2^{j-2m} \cdot \frac{d_t^{2j-2m}}{R_t^{2j-2m+2\tau}}.$$

Inserting this into $\mathcal{Q}_j^\tau(t)$, we obtain

$$\begin{aligned}
\mathcal{Q}_j^\tau(t) &= \sum_{m=0}^{\lfloor \frac{j}{2} \rfloor} \frac{(-1)^m \cdot (\tau)_{j-m}}{(j-2m)! m!} \cdot 2^{j-2m} \cdot \frac{\mu_t^{j-2m}}{R_t^{2\tau}} \cdot \left(\frac{d_t}{R_t}\right)^{2j-2m} \\
&= \frac{1}{R_t^{2\tau}} \cdot \left(\frac{d_t}{R_t}\right)^j \cdot \sum_{m=0}^{\lfloor \frac{j}{2} \rfloor} \frac{(-1)^m \cdot (\tau)_{j-m}}{(j-2m)! m!} \cdot (2v_t)^{j-2m}
\end{aligned}$$

with $v_t = \mu_t \frac{d_t}{R_t}$, so that $|v_t| \leq 1$. Theorem C.4 then yields

$$\mathcal{Q}_j^\tau(t) = \frac{1}{R_t^{2\tau}} \cdot \left(\frac{d_t}{R_t}\right)^j \cdot \mathcal{C}_j^\tau(v_t)$$

and therefore

$$\begin{aligned}
|\mathcal{Q}_j^\tau(t)| &\leq \frac{1}{R_t^{2\tau}} \cdot \left(\frac{d_t}{R_t}\right)^j \cdot |\mathcal{C}_j^\tau(v_t)| \\
&\leq \frac{1}{R_t^{2\tau}} \cdot \left(\frac{d_t}{R_t}\right)^j \cdot \frac{(2\tau)_j}{j!}.
\end{aligned}$$

□

This bounding of $\mathcal{Q}_j^\tau(t)$ paves the way for an estimation of the error kernel $\kappa_{(p+1)}^\tau$ of the function G_σ^τ.

Theorem 3.2.5
For all $p \in \mathbb{N}_0$ and $|x - \hat{x}| = d > c = |y - \hat{x}|$, the error kernel $\kappa^\tau_{(p+1)}$ is bound by

$$|\kappa^\tau_{(p+1)}| < \mathcal{D}_{2\tau-1}(p+1) \cdot \frac{1}{(d-c)^{2\tau}} \cdot \left(\frac{c}{d-c}\right)^{p+1},$$

where $\mathcal{D}_{2\tau-1}(s)$ is a polynomial in s of degree $2\tau - 1 \in \mathbb{N}_0$.

Proof. Using Definition 3.2.2, the error kernel $\kappa^\tau_{(p+1)}$ of G^τ_σ is defined by

$$\kappa^\tau_{(p+1)} = (p+1) \int_0^1 (1-t)^p \left(\frac{c}{d_t}\right)^{p+1} \cdot \mathcal{Q}^\tau_{p+1}(t)\,\mathrm{d}t$$

for $p \in \mathbb{N}_0$. We apply Theorem 3.2.4 and obtain

$$\begin{aligned}\kappa^\tau_{(p+1)} &= (p+1) \int_0^1 (1-t)^p \left(\frac{c}{R_t}\right)^{p+1} \frac{1}{R_t^{2\tau}} \cdot \mathcal{C}^\tau_{p+1}(v_t)\mathrm{d}t \\ &= (p+1) \cdot \frac{c^{p+1}}{d^{p+1+2\tau}} \cdot \int_0^1 \mathcal{C}^\tau_{p+1}(v_t) \cdot (1-t)^p \left(\frac{d}{R_t}\right)^{p+1+2\tau} \mathrm{d}t.\end{aligned}$$

The same theorem gives us then

$$\begin{aligned}|\kappa^\tau_{(p+1)}| &\le (p+1) \cdot \frac{c^{p+1}}{d^{p+1+2\tau}} \cdot \frac{(2\tau)_{p+1}}{(p+1)!} \cdot \int_0^1 (1-t)^p \left(\frac{d}{R_t}\right)^{p+1+2\tau} \mathrm{d}t \\ &=: \mathcal{D}_{2\tau-1}(p+1) \cdot (p+1) \cdot \frac{c^{p+1}}{d^{p+1+2\tau}} \cdot \int_0^1 (1-t)^p \left(\frac{d}{R_t}\right)^{p+1+2\tau} \mathrm{d}t,\end{aligned}$$

where

$$\mathcal{D}_{2\tau-1}(p+1) := \frac{(2\tau)_{p+1}}{(p+1)!} = \frac{(p+2)_{2\tau-1}}{\Gamma(2\tau)}$$

is a polynomial in $p+1$ of degree $2\tau - 1$ if $2\tau - 1 \in \mathbb{N}_0$.
We define

$$I(p+1) := (p+1) \int_0^1 (1-t)^p \left(\frac{d}{R_t}\right)^{p+1+2\tau} \mathrm{d}t.$$

With Definition 3.2.2 of d_t we can write

$$d_t^2 = (x - \hat{x} - t(y - \hat{x}))^T (x - \hat{x} - t(y - \hat{x})) = d^2 - 2(x - \hat{x})^T (y - \hat{x})t + c^2 t^2$$

and therefore

$$\begin{aligned}\frac{d_t^2}{d^2} &= 1 - 2\mu\alpha t + \alpha^2 t^2 =: \tilde{d}_t^2(\mu),\\ \mu &:= \frac{(x-\hat{x})^T(y-\hat{x})}{dc},\\ \alpha &:= \frac{c}{d}.\end{aligned}$$

We have $|\mu| < 1$ and therefore $\tilde{d}_t^2(\mu) > \tilde{d}_t^2(1)$ for fixed t and α, so that

$$\frac{R_t^2}{d^2} = \frac{d_t^2 + \sigma^2}{d^2} > \tilde{d}_t^2(1) + \frac{\sigma^2}{d^2} =: \tilde{d}_t^2(1) + \beta^2.$$

With $d_t^2(1) = (1-\alpha t)^2$ this yields

$$\left(\frac{d}{R_t}\right)^{p+1+2\tau} < \frac{1}{((1-\alpha t)^2 + \beta^2)^{\frac{p+1+2\tau}{2}}} \leq \frac{1}{(1-\alpha t)^{p+1+2\tau}}$$

for $\sigma \geq 0$. We insert this estimation into $I(p+1)$ and obtain with Theorem C.6

$$I(p+1) < (p+1)\int_0^1 \frac{(1-t)^p}{(1-\alpha t)^{p+1+2\tau}}\,\mathrm{d}t = H(1, p+1+2\tau; p+2; \alpha),$$

since we assumed $\alpha < 1$. Using Definitions C.3, C.5 and Theorems C.2, C.4 this leads to

$$\begin{aligned}I(p) &< \sum_{n=0}^{\infty} \frac{(1)_n \cdot (p+1+2\tau)_n}{(p+2)_n} \cdot \frac{\alpha^n}{n!}\\ &= \sum_{n=0}^{\infty} (p+1+2\tau)_n \cdot \frac{\Gamma(p+2)}{\Gamma(p+2+n)} \cdot \alpha^n\\ &\leq \sum_{n=0}^{\infty} (p+1+2\tau)_n \cdot \frac{\Gamma(p+2)}{\Gamma(p+2)\Gamma(n+1)} \cdot \alpha^n\\ &= \sum_{n=0}^{\infty} \frac{(p+1+2\tau)_n}{n!} \cdot \alpha^n\\ &= \sum_{n=0}^{\infty} \mathcal{C}_n^{\frac{p+1+2\tau}{2}}(1) \cdot \alpha^n = \frac{1}{(1-\alpha)^{p+1+2\tau}}.\end{aligned}$$

We now return to the estimation of $\kappa^{\tau}_{(p+1)}$ and insert our estimate of $I(p+1)$. We finally obtain

$$\begin{aligned}|\kappa^{\tau}_{(p+1)}| &\leq \mathcal{D}_{2\tau-1}(p+1) \cdot \frac{c^{p+1}}{d^{p+1+2\tau}} \cdot I(p+1)\\ &< \mathcal{D}_{2\tau-1}(p+1) \cdot \frac{c^{p+1}}{d^{p+1+2\tau}} \cdot \frac{1}{(1-\alpha)^{p+1+2\tau}}\\ &= \mathcal{D}_{2\tau-1}(p+1) \cdot \frac{1}{(d-c)^{2\tau}} \cdot \left(\frac{c}{d-c}\right)^{p+1}.\end{aligned}$$

□

Remark 3.2.6
It is worth noting that the estimate

$$\frac{1}{((1-\alpha t)^2+\beta^2)^{\frac{p+1+2\tau}{2}}} \leq \frac{1}{(1-\alpha t)^{p+1+2\tau}}$$

is actually not necessary. In [92], we find an explicit formulation of the integral with β^2 included. However, as this involves generalized hypergeometric functions with nontrivial arguments, we cannot expect to find such a closed form for $|\kappa^\tau_{(p+1)}|$ as stated in the present result. Consequently, the estimation is decoupled from the core radius σ, which might leave room for further optimizations.

This result can be already used to show convergence of the multipole expansion. However, the following section shows that we can extend the convergence region significantly by incorporating the core radius σ.

3.2.3 Convergence of the multipole expansion

If we take Theorem 3.2.5 as a basis for convergence analysis, we can state

$$d > 2c \Rightarrow \lim_{p\to\infty} \kappa^\tau_{(p+1)}(x,y) = 0.$$

If this relation were true in the opposite direction, then the condition $d > 2c$ would be not only sufficient but also necessary, limiting the convergence region of the series expansion significantly. However, the following theorem shows that this relation cannot be reversed, i.e. that the convergence condition can be improved further. We again adapt the following result from [90].

Theorem 3.2.7
For all $\tau > 0$ and $y \in \mathcal{V}$ we have

$$\lim_{p\to\infty} \kappa^\tau_{(p+1)}(x,y) = 0 \text{ for all } x \in U_y$$

with $U_y := \{x : |x-\hat{x}| > \sqrt{|y-\hat{x}|^2-\sigma^2}\}$.

Proof. We return to the estimation of $\kappa^\tau_{(p+1)}$ in the proof of Theorem 3.2.5, stating

$$|\kappa^\tau_{(p+1)}| \leq \mathcal{D}_{2\tau-1}(p+1)\cdot(p+1)\cdot\frac{c^{p+1}}{d^{p+1+2\tau}}\cdot\int_0^1 (1-t)^p \left(\frac{d}{R_t}\right)^{p+1+2\tau} \mathrm{d}t.$$

To analyze this expression for increasing p, we define

$$K(p+1) := \mathcal{D}_{2\tau-1}(p+1) \cdot (p+1) \cdot \left((1-t) \cdot \frac{\alpha d}{R_t}\right)^p,$$

omitting terms which do not depend on p. Obviously, we have

$$\lim_{p\to\infty} K(p+1) = 0 \text{ for } (1-t) \cdot \frac{\alpha d}{R_t} < 1.$$

Using

$$\frac{d_t^2}{d^2} = 1 - 2\mu\alpha t + \alpha^2 t^2$$

we have to verify

$$(1-t)^2\alpha^2 < 1 - 2\mu\alpha t + \alpha^2 t^2 + \beta^2$$

or $\alpha^2 < 1 + \beta^2$, since $|\mu| < 1$. This is clearly ensured, if $d > \sqrt{c^2 - \sigma^2}$. □

The next result detaches this statement from the actual $y \in \mathcal{V}$. Finally, this yields a convergence and error term statements for the interaction between a particle located at x and the multipole origin $\hat{x}$ of a cluster $\mathcal{V}$ in contrast to direct interactions between x and all members y of $\mathcal{V}$.

Theorem 3.2.8 (Convergence Theorem for multipole expansions)
The multipole expansion for $G_\sigma^\tau \in D_\sigma$ with

$$G_\sigma^\tau(x) = \frac{1}{(|x|^2 + \sigma^2)^\tau}$$

converges for all $\tau > 0$ and all x outside a ball with radius $\sqrt{b^2 - \sigma^2}$, centered on $\hat{x}$, with $b := \sup\{|y - \hat{x}| : y \in \mathcal{V}\}$, i.e. we have

$$\lim_{p\to\infty} \varepsilon_{(p+1)}^\tau(x) = 0 \text{ for all } x \in U$$

with $U := \{x : |x - \hat{x}| = d > \sqrt{b^2 - \sigma^2}\}$.
Furthermore, for $d > b$ an explicit error bound on $\varepsilon_{(p+1)}^\tau$ is given by

$$\left|\varepsilon_{(p+1)}^\tau(x)\right| \le \mathcal{D}_{2\tau-1}(p+1) \cdot \frac{\bar{M}_{(0)}}{(d-b)^{2\tau}} \cdot \left(\frac{b}{d-b}\right)^{p+1},$$

where

$$\bar{M}_{(0)} := \sum_{y\in\mathcal{V}} |\alpha_y|,$$

$$\mathcal{D}_{2\tau-1}(s) := \frac{(2\tau)_s}{s!}.$$

Proof. With $c = |y - \hat{x}| \leq \sup\{|y - \hat{x}| : y \in \mathcal{V}\} = b$ and

$$\left|\varepsilon^{\tau}_{(p+1)}(x)\right| \leq \sum_{y \in \mathcal{V}} |\kappa^{\tau}_{(p)}(x, y)| \cdot |\alpha_y|$$

this result follows directly from Theorem 3.2.7. □

Remark 3.2.9
Besides the region of convergence, this theorem gives us the convergence rate, at least for $d > 2b$. This rate decreases for increasing τ, because the degree of the polynomial $\mathcal{D}_{2\tau-1}(s)$ increases. However, the speed of convergence is determined by a geometric sequence whose basis is controlled by the ratio of cluster size and distance to this cluster. Some examples are shown in the next section.

This Convergence Theorem already covers multipole expansions for a huge range of kernels. While the error estimate is independent of σ, the convergence region is clearly influenced by this parameter. Before we generalize this result further by applying it to algebraic smoothing kernels and therefore vortex particle methods, we discuss our result and place it into the context of other results covering similar $\rho^{-\tau}$-kernels.

3.2.4 Discussion and implications

Both, the convergence result and error analysis of the previous section are based on the approach of Salmon [90] and generalize the ideas therein, covering not only the (softened) ρ^{-1}-kernel but also all members of the classes D_σ. Since the (softened) ρ^{-1}-kernel is a member of D_σ with core size $\sigma \geq 0$ and $\tau = 1/2$ the present theory can be seen as a substantial extension of Salmon's concept.

However, it should be mentioned that a direct application of our theory to this particular kernel does not reproduce the error bound of [90], which reads in our terms (see Definition 3.2.2)

$$\left|\varepsilon^{\frac{1}{2},\mathrm{S}}_{(p+1)}(x)\right| \leq \frac{\bar{M}_{(0)}}{d-b} \cdot \left(\frac{b}{d}\right)^{p+1}.$$

Here, the denominator of the last factor does not depend on the cluster extension b, causing a slightly better convergence behavior in multipole order p for fixed d and b as well as a more rapid decay of the error for fixed p and b but increasing d. The reason for this difference can be found in the estimation of $I(p+1)$ in the proof of Theorem 3.2.5. Only for the particular value $\tau = 1/2$ we can write

$$\sum_{n=0}^{\infty} \frac{(1)_n \cdot (p+1+2\tau)_n}{(p+2)_n} \cdot \frac{\alpha^n}{n!} = \sum_{n=0}^{\infty} \alpha^n = \frac{1}{1-\alpha},$$

detaching α from the p-dependency. This reduction seems to be impossible for general $\tau > 0$, making the case $\tau = 1/2$ a very special one, which is a well-known conclusion in the field of Fast Multipole Methods as well [93].

Although in explicit form, the drawback of Salmon's error bound is its non-linearity in d. For a numerical implementation one can use error bounds as a criterion to decide whether or not an interaction between a particle located at x and a cluster $\mathcal{V}$ with multipole origin $\hat{x}$ and distance d is acceptable regarding its induced approximation error. If not, the cluster is rejected and its sub-clusters are examined. These so-called 'multipole acceptance criteria' (MAC) are crucial for the development of multipole-based tree codes, especially for Barnes-Hut tree codes [82, 94, 95]. Using error bounds as MAC leads to so-called 'data-dependent' MAC, i.e. criteria taking into account the actual particle and charge distribution inside a cluster and not only its size or location [55, 95]. However, an error bound with non-linear dependence on d cannot be rewritten explicitly in terms of d, which makes their evaluation very expensive. In contrast to Salmon's error bound, our result of Theorem 3.2.8 can be easily rewritten to act as a data dependent MAC for any given user-specified error per interaction $\Delta\varepsilon$. Demanding

$$|\varepsilon^{\tau}_{(p+1)}(x)| \leq \mathcal{D}_{2\tau-1}(p+1) \cdot \frac{\bar{M}_{(0)}}{(d-b)^{2\tau}} \cdot \left(\frac{b}{d-b}\right)^{p+1} \leq \Delta\varepsilon,$$

we can write

$$d \geq b + \left(\frac{\mathcal{D}_{2\tau-1}(p+1) \cdot \bar{M}_{(0)} \cdot b^{p+1}}{\Delta\varepsilon}\right)^{\frac{1}{p+1+2\tau}},$$

so that for all d fulfilling this inequality the interaction between particle and multipole induces a maximal error of $\Delta\varepsilon$.

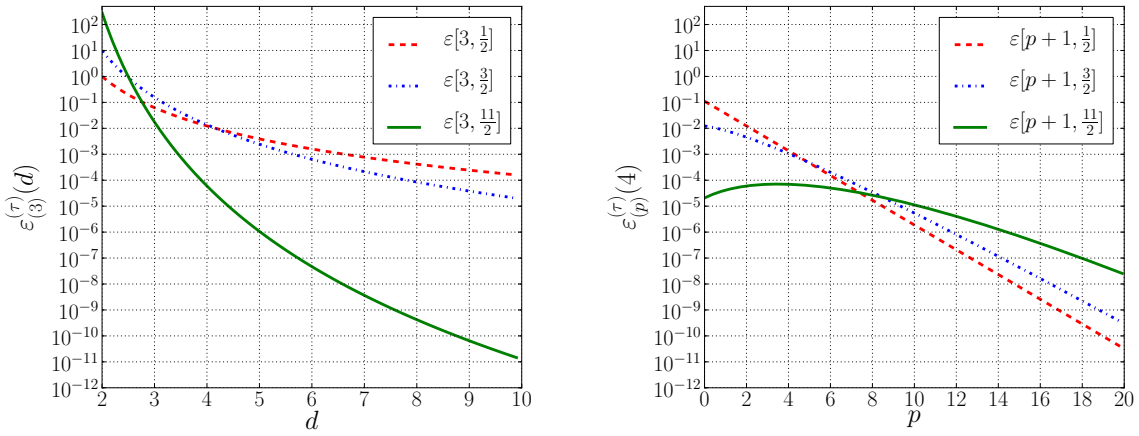
Furthermore, the Convergence Theorem motivates another, much simpler and faster multipole acceptance criterion. Here, the maximum distance between multipole origin and all members of the cluster – which is b in our notation – is compared to the distance between x and the multipole origin $\hat{x}$ – denoted d in our theory (see Definition 3.2.2 for notation). A tunable and user-defined parameter θ controls the ratio b/d, so that the multipole expansion is accepted if and only if

$$\frac{b}{d} < \theta.$$

This MAC is commonly called 'Bmax-MAC' and more on this topic can be found in [95] and Section 4.1.3.

To get an impression of the behavior of the error terms

$$\begin{aligned}\left|\varepsilon^{\tau}_{(p+1)}(x)\right| &\leq \mathcal{D}_{2\tau-1}(p+1) \cdot \frac{\bar{M}_{(0)}}{(d-b)^{2\tau}} \cdot \left(\frac{b}{d-b}\right)^{p+1} \\ &= \frac{(2\tau)_{p+1}}{(p+1)!} \cdot \frac{\bar{M}_{(0)}}{(d-b)^{2\tau}} \cdot \left(\frac{b}{d-b}\right)^{p+1}\end{aligned}$$



(a) Error terms for increasing distance d with $b = 1$ and $p = 2$.

(b) Error terms for increasing order p with $d = 4$ and $b = 1$.

Figure 3.3: Visualization of the error terms $\varepsilon[p+1, 1/2]$, $\varepsilon[p+1, 3/2]$ and $\varepsilon[p+1, 11/2]$ from Theorem 3.2.8 for the corresponding kernels G_0^τ. For abbreviation we define $\varepsilon^\tau_{(p+1)}(x) = \varepsilon[p+1, \tau]$. We choose $\sigma = 0$ and all other constants such as $\bar{M}_{(0)}$ as one. In (a), quadrupole order $p = 2$ reflects the common usage in Barnes-Hut tree codes, while in (b) $d = 4$ is an average value for the distance to a cluster.

of Theorem 3.2.8, we visualize them for different kernels in Figure 3.3. We choose $\tau = 1/2$, $\tau = 3/2$ and $\tau = 11/2$, covering different components as they appear in the decomposition of the regularization kernels. Furthermore, we set $\sigma = 0$, since the error estimate does not depend on σ, so that the corresponding kernels are

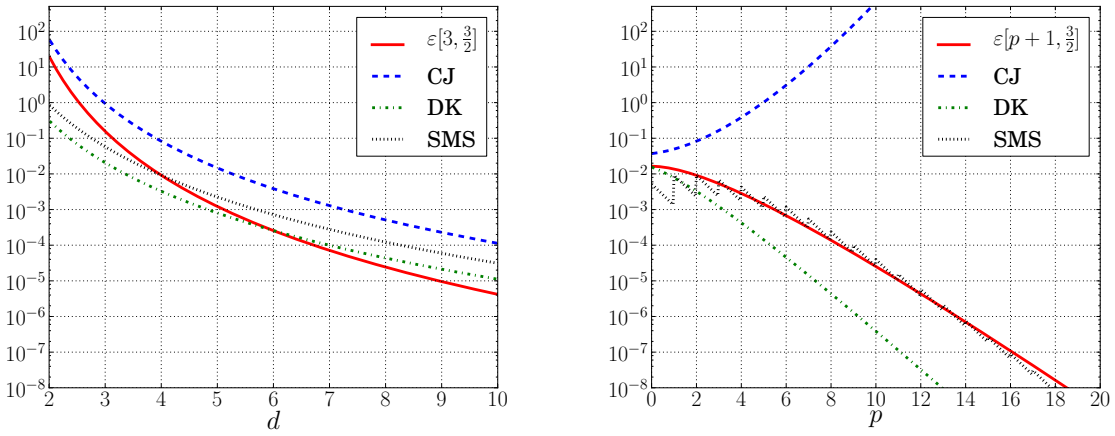
$$G_0^{\frac{1}{2}}(\rho) = \frac{1}{\rho}, \quad G_0^{\frac{3}{2}}(\rho) = \frac{1}{\rho^3} \quad \text{and} \quad G_0^{\frac{11}{2}}(\rho) = \frac{1}{\rho^{11}}.$$

All other constants such as $\bar{M}_{(0)}$ are fixed to one for the sake of simplicity. For abbreviation we define $\varepsilon^\tau_{(p+1)}(x) = \varepsilon[p+1, \tau]$. The choice $p = 2$ reflects the common usage in tree codes, while $d = 4$ is an average value for the distance to a cluster [55, 94, 95]. For increasing τ the rate of convergence tends to be significantly better for larger d with fixed p, but with increasing p and fixed d low numbers of τ provide a slightly better rate.

To compare our result, we cite three different approaches and error bounds and compare them to Theorem 3.2.8. We reformulate these results to match our notation and again set all constants to one:

a) Chowdhury and Jandhyala [96]

$$\left|\varepsilon^{\tau,\mathrm{CJ}}_{(p+1)}(x)\right| \leq \frac{1}{d^\tau} \cdot \frac{(\tau)_{p+1}}{\left(1 - \frac{b}{d}\right)^{\tau+p+1}} \cdot \left(\frac{b}{d}\right)^{p+1},$$



(a) Comparison of error terms for increasing distance d with $b = 1$, $p = 2$, $\tau = \frac{3}{2}$.

(b) Comparison of error terms for increasing multipole order p with $b = 1$, $d = 4$, $\tau = \frac{3}{2}$.

Figure 3.4: Comparison of other error terms taken from [96, 97, 98], exemplarily for $G_0^{\frac{3}{2}}$. Again, we choose $\sigma = 0$ and all other emerging constants like total charge or cluster size as one.

b) Duan and Krasny [97]

$$\left|\varepsilon_{(p+1)}^{\tau,\mathrm{DK}}(x)\right| \leq \frac{1}{d^\tau} \cdot \frac{1}{\Gamma(\tau)} \cdot \frac{\mathrm{d}^{\tau-1}}{\mathrm{d}s^{\tau-1}} \left(\frac{s^{p+\tau}}{1-s}\right)\Bigg|_{s=\frac{b}{d}},$$

c) Srinivasan, Mahawar and Sarin [98]

$$\left|\varepsilon_{(p+1)}^{\tau,\mathrm{SMS}}(x)\right| \leq \frac{1}{d^\tau} \cdot \frac{B}{\left(1-\frac{b}{d}\right)} \cdot \left(\frac{b}{d}\right)^{p+1},$$

$$B = \sum_{n=0}^{p} \sum_{m=0}^{\lfloor \frac{n}{2} \rfloor} \left| \frac{(\tau)_{n-m} \cdot \left(\tau - \frac{1}{2}\right)_m}{\left(\frac{3}{2}\right)_{n-m} \cdot m!} \cdot (2n-4m+1) \right|.$$

While the first error term (CJ) basically has the same shape like ours – and it is derived in a similar way – both the second (DK) and third term (SMS) lack an explicit, closed form, so that the numerical derivation necessary for data dependent MACs is much more expensive, making them unfavorable in this case.

Figure 3.4 shows the comparison of the different approaches for

$$G_0^{\frac{3}{2}} = \frac{1}{\rho^3}.$$

Here, $\sigma = 0$ and all other emerging constants like total charge or cluster size are set to one. For $p = 2$ all competing error terms are inferior for $d > 6$, at least for this particular example. However, with fixed distance d and increasing multipole order p, we obtain a different result. While the CJ-error term lacks convergence at all, the other two error bounds perform better for $p > 14$. Especially the DK-formula shows its strength already for small values of p. Nevertheless, since there seems to be no closed form at hand for this error bound, our estimation is still advantageous in this case. For different values of τ the observations above are roughly the same, but with slightly different thresholds for p and d.

We can conclude that the resulting error bound of Theorem 3.2.8 is in good agreement with already existing analyses in [96, 97, 98] for $\rho^{-\tau}$-kernels, but with the advantages of extending the multipole expansion theory to softened kernels and deriving an error bound in explicit shape. In the following section, we generalize this result to the class A_σ of algebraic smoothing kernels by making use of the Decomposition Theorem and extend it further to cover field derivatives needed in vortex particle methods.

3.2.5 Application to vortex particle methods

The rate of convergence as well as the error bound for a combined smoothing kernel $G_\sigma \in A_\sigma$ are determined by the linear combination of the G_σ^τ, e.g. with

$$G_\sigma(x) = \sum_{l=0}^{\lambda} b_l \, G_\sigma^{l+\frac{1}{2}}(x),$$

with adequate choices of coefficients b_l. The convergence condition of Theorem 3.2.8 applies here as well, but an explicit error bound depending only on λ and p cannot be stated in general. The reason for this drawback is the $(d-b)^{2l+1}$ term in the denominator. We have to distinguish between

$$\left|\varepsilon_{(p+1)}(x)\right| \leq \tilde{\mathcal{D}}_{2\lambda}(p+1) \cdot \frac{\bar{M}_{(0)}}{(d-b)^{2\lambda+1}} \cdot \left(\frac{b}{d-b}\right)^{p+1} \quad \text{for } d-b<1,$$

and

$$\left|\varepsilon_{(p+1)}(x)\right| \leq \tilde{\mathcal{D}}_{2\lambda}(p+1) \cdot \frac{\bar{M}_{(0)}}{(d-b)} \cdot \left(\frac{b}{d-b}\right)^{p+1} \quad \text{for } d-b>1,$$

where

$$\tilde{\mathcal{D}}_{2\lambda}(p+1) := \sum_{l=0}^{\lambda} |a_{\lambda-l}^{(l)}| \cdot \sigma^{2l} \cdot \mathcal{D}_{2l}(p+1)$$

is a polynomial in p of degree 2λ.

We continue our previous Example 3.1.13 of the second and sixth order algebraic kernels $G_\sigma^{(2)}$ and $G_\sigma^{(6)}$. Applying the Decomposition Theorem to obtain explicit kernels, we can circumvent the aforementioned drawback.

Example 3.2.10

a) The decomposition of $G_\sigma^{(2)}$ into $G_\sigma^{\frac{1}{2}}$ and $G_\sigma^{\frac{3}{2}}$, as illustrated in Example 3.1.13, yields

$$\left|\varepsilon_{(p+1)}^{\frac{1}{2}}(x)\right| \leq \mathcal{D}_0(p+1) \cdot \frac{\bar{M}_{(0)}}{d-b} \cdot \left(\frac{b}{d-b}\right)^{p+1} = \frac{\bar{M}_{(0)}}{d-b} \cdot \left(\frac{b}{d-b}\right)^{p+1},$$
$$\left|\varepsilon_{(p+1)}^{\frac{3}{2}}(x)\right| \leq \frac{(p+2)(p+3)}{2} \cdot \frac{\bar{M}_{(0)}}{(d-b)^3} \cdot \left(\frac{b}{d-b}\right)^{p+1},$$

so that the linear combination of $G_\sigma^{\frac{1}{2}}$ and $G_\sigma^{\frac{3}{2}}$ with $b_0 = \frac{1}{4\pi}$ and $b_1 = \frac{\sigma^2}{8\pi}$ leads to

$$\left|\varepsilon_{(p+1)}(x)\right| \leq \frac{\bar{M}_{(0)}}{4\pi(d-b)} \cdot \left(\frac{b}{d-b}\right)^{p+1} \cdot \left(1 + \frac{(p+2)(p+3)}{4} \cdot \frac{\sigma^2}{(d-b)^2}\right).$$

b) Furthermore, we have

$$G_\sigma^{(6)}(\rho) = \sum_{l=0}^{5} b_l G_\sigma^{l+\frac{1}{2}}(\rho)$$

and therefore

$$|\varepsilon_{(p+1)}(x)| \leq \frac{\bar{M}_{(0)}}{d-b} \cdot \left(\frac{b}{d-b}\right)^{p+1} \cdot \left(\frac{128}{512\pi} + \sum_{l=1}^{5} \frac{a_l}{\Gamma(2l+1)} \prod_{j=0}^{l} (p+2+j) \frac{\sigma^{2l}}{(d-b)^{2l}}\right),$$

where a_l are the coefficients of $G_\sigma^{l+\frac{1}{2}}$.

It is worth noting that the whole theory can also be applied to the rezoning concept described in Section 2.2.1 and even to the particle strength exchange (PSE) scheme that has been introduced in Section 2.2.3. The rezoning approach recomputes the vorticity after a few time steps on new particle locations y_q, $q \in \mathcal{M}$, $|\mathcal{M}| = M$, with the well-known formula

$$\omega^{\text{mesh}}(y_q) = \sum_{p \in \mathcal{P}} \zeta_\sigma(y_q - x_p) \cdot \omega^{\text{part}}(x_p) \text{vol}_p,$$

i.e. using the standard regularized smoothing kernel ζ_σ and the given vorticity values $\omega^{\text{part}}(x_p)$. Since of course $\zeta_\sigma \in A_\sigma$, this approach can be directly implemented with a multipole-based fast summation method, supported by the theory of this work. The computational costs in comparison to a direct summation are then reduced from $\mathcal{O}(NM)$ to at least $\mathcal{O}(M \log N)$.

To see that even PSE can be handled by this theory, we recall the central idea of this approach. The diffusion operator Δ is replaced by an integral operator, which is then discretized using particles, yielding

$$\Delta\omega(x) \approx \tilde{\Delta}_\sigma\omega(x) = \frac{1}{\sigma^2}\sum_{q\in\mathcal{P}'}\eta_\sigma(x-x_q)\cdot(\omega(x_q)-\omega(x))\mathrm{vol}_q.$$

We have seen that one convenient way of defining this approximation kernel η is by means of the smoothing kernel ζ with

$$\eta(x) := -2\,\frac{\nabla\zeta(x)\cdot x}{|x|^2}.$$

Again, the Decomposition Theorem is of great help here.

Theorem 3.2.11
For $\zeta^\lambda \in A_1$ and $\lambda \in \mathbb{N}_0$, we obtain $\eta^{\lambda'} \in A_1$ with $\lambda' = \lambda + 1$.

Proof. Without loss of generality, we can restrict ourselves to $\zeta^\tau \in D_1$, since following Theorem 3.1.5 each $\zeta_\sigma \in A_1$ is a linear combination of functions of D_1. For $|x| = \rho$ and

$$\zeta^\tau(\rho) = \frac{1}{(\rho^2+1)^\tau}$$

we have

$$\nabla\zeta^\tau(x) = -x\cdot\frac{2\tau}{(|x|^2+1)^{\tau+1}}$$

and therefore

$$\eta^{\tau'}(x) = -2\,\frac{\nabla\zeta^\tau(x)\cdot x}{|x|^2} = 4\tau\,\frac{x\cdot x}{|x|^2\cdot(|x|^2+1)^{\tau+1}} = \frac{4\tau}{(|x|^2+1)^{\tau+1}}.$$

Therefore, $\eta^{\tau'} \in D_1$ with $\tau' = \tau + 1$, so that a combined approximation kernel is again a member of A_1. □

This short theorem shows that even PSE can profit from the theory presented. Discretizing the integral operator with M quadrature particles, the complexity of this method is reduced from $\mathcal{O}(NM)$ to at least $\mathcal{O}(N\log M)$. However, the multipole moments are very different from the ones we already introduced in Definition 3.2.1. The coefficients do not only depend on the vorticity $\omega(x_q)$ and the volume vol_q of the quadrature points but also on the vorticity $\omega(x)$ at the target point, which does not affect the theoretical complexity of this approach but its implementation.

Since we now have a closed multipole expansion theory for algebraic smoothing functions at hand, the regularized and discretized vorticity $\tilde{\omega}_\sigma$ and stream function $\tilde{\psi}_\sigma$ with

$$\tilde{\omega}_\sigma(x) = \sum_{p\in\mathcal{P}} \zeta_\sigma(x - x_p)\cdot\alpha_p,$$
$$\tilde{\psi}_\sigma(x) = \sum_{p\in\mathcal{P}} G_\sigma(x - x_p)\cdot\alpha_p$$

and even rezoning and PSE can be computed using a multipole-based fast summation method. However, as we have seen in Algorithm 2.1, vortex particle methods require the evaluation of the velocity

$$u_\sigma(x_p) = \sum_{q\in\mathcal{P}} \nabla G_\sigma(x_p - x_q)\times\alpha_q$$

and the inviscid vorticity update

$$\frac{\mathrm{d}\alpha_p}{\mathrm{d}t} = (\alpha_p\cdot\nabla)\,\tilde{u}_\sigma(x_p).$$

In each of these two equations the initial regularized stream function kernel G_σ is modified by a differential operator. Using the Decomposition Theorem, these operators are applied to the components G^τ_σ of G_σ, as we have already seen in Theorem 3.2.11. The following theorem shows that the general convergence criterion of Theorem 3.2.8 is not affected by these modifications.

Theorem 3.2.12
The multipole expansions for ∇G^τ_σ and $\nabla^2 G^\tau_\sigma$ with $G^\tau_\sigma \in D_\sigma$ converge for all $\tau > 0$ and all x outside a ball with radius $\sqrt{b^2-\sigma^2}$, centered on $\hat{x}$, with $b := \sup\{|y-\hat{x}| : y\in\mathcal{V}\}$.

Proof. In this proof we use the same idea as in Theorem 3.2.5. Starting from the expression

$$\kappa^\tau_{(p+1)} = (p+1)\cdot\frac{c^{p+1}}{d^{p+1+2\tau}}\cdot\int_0^1 \mathcal{C}^\tau_{p+1}(v_t)\cdot(1-t)^p\left(\frac{d}{R_t}\right)^{p+1+2\tau}\mathrm{d}t,$$

we obtain by differentiation

$$\nabla\kappa^\tau_{(p+1)} = (p+1)\cdot c^{p+1}\cdot\int_0^1 (1-t)^p\cdot\nabla\left(\frac{\mathcal{C}^\tau_{p+1}(v_t)}{R_t^{p+1+2\tau}}\right)\mathrm{d}t,$$

where

$$\nabla\left(\frac{\mathcal{C}^\tau_{p+1}(v_t)}{R_t^{p+1+2\tau}}\right) = -\frac{(p+1+2\tau)}{R_t^{p+2+2\tau}}\cdot\mathcal{C}^\tau_{p+1}(v_t)\cdot\frac{(x-\hat{x}-t(y-\hat{x}))}{R_t}$$
$$+\frac{1}{R_t^{p+1+2\tau}}\frac{y-\hat{x}}{c}\cdot\left(1-\frac{d_t^2}{R_t^2}\right)\frac{1}{R_t}\left(\frac{\mathrm{d}}{\mathrm{d}s}\mathcal{C}^\tau_{p+1}(s)\right)\Big|_{s=v_t}.$$

With Theorem C.4 we have

$$\left|\nabla\left(\frac{\mathcal{C}^\tau_{p+1}(v_t)}{R_t^{p+1+2\tau}}\right)\right| \leq \frac{1}{R_t^{p+2+2\tau}}\left((p+1+2\tau)\cdot\frac{(2\tau)_{p+1}}{(p+1)!}+2\tau\cdot\frac{(2\tau+2)_p}{p!}\right)$$
$$= \frac{2\tau}{R_t^{p+2+2\tau}}\frac{(2\tau+2)_{p+1}}{(p+1)!}.$$

Therefore,

$$\left|\nabla\kappa^\tau_{(p+1)}\right| \leq \tilde{\mathcal{D}}_{2\tau+1}(p+2)\cdot(p+1)\cdot c^{p+1}\cdot\int_0^1(1-t)^p\frac{1}{R_t^{p+2+2\tau}}\mathrm{d}t,$$

where

$$\tilde{\mathcal{D}}_{2\tau+1}(p+2) := 2\tau\frac{(2\tau+2)_{p+1}}{(p+1)!} = \frac{2\tau}{\Gamma(2\tau+2)}\cdot(p+2)_{2\tau+1}$$

is a polynomial in $p+2$ of degree $2\tau+1\in\mathbb{N}_0$. Following the proof of 3.2.5, we obtain

$$\left|\nabla\kappa^\tau_{(p+1)}\right| < \tilde{\mathcal{D}}_{2\tau+1}(p+2)\cdot\frac{1}{(d-c)^{2\tau+1}}\cdot\left(\frac{c}{d-c}\right)^{p+1},$$

yielding a comparable error bound for the expansion of ∇G^τ_σ. Moreover, the procedure of the proof of Theorem 3.2.7 is directly applicable to the estimation of $\nabla\kappa^\tau_{(p+1)}$, so that

$$\lim_{p\to\infty}\nabla\kappa^\tau_{(p+1)}(x,y) = 0 \text{ for all } x\in U_y$$

with $U_y := \{x : |x-\hat{x}| > \sqrt{|y-\hat{x}|^2-\sigma^2}\}$.
The same approach is valid for the expansion of $\nabla^2 G^\tau_\sigma$. Multiple applications of differential formulas give

$$\left|\nabla^2\kappa^\tau_{(p+1)}\right| < \tilde{\mathcal{E}}(p)\cdot\frac{1}{(d-c)^{2\tau+2}}\cdot\left(\frac{c}{d-c}\right)^{p+1}$$

with a suitable polynomial $\tilde{\mathcal{E}}(p)$. □

Remark 3.2.13
In both cases a direct statement of an error bound for the corresponding error terms is not preferable since the operands coupling kernel derivatives and multipole moments, i.e. cross-product and vorticity-vector, are non-trivial. However, from an algorithmic point of view, both expansions use only permutations of simple multipole moments

$$M^{i_1\ldots i_j}_{(j)} = \sum_{y_p\in\mathcal{V}}(y_p-\hat{x})^{i_1}\cdot\ldots\cdot(y_p-\hat{x})^{i_j}\cdot\alpha_p,$$

as given by Definition 3.2.1. Having been precomputed, these moments can be used for the derivation of stream function, velocity and vorticity update. Again, we note that this is not valid for a multipole-based implementation of PSE.

Common implementations of multipole-based tree codes for regularized vortex particle methods often make use of the second order Gaussian kernel and its stream function kernel [55, 99]

$$G(\rho) := \frac{1}{4\pi\rho} \operatorname{erf}\left(\frac{\rho}{\sqrt{2}}\right).$$

Two possible ways are considered: In [99] the Coulomb-potential ρ^{-1} is used as an approximation of $G(\rho)$ for $\rho > 4$, since the multipole expansion theory is well-known in that case. This procedure requires direct interactions within the area of $\rho \leq 4$, where the multipole expansion is not valid. In [55], a full expansion of the Gaussian kernel is implemented, at least for $\rho > 5$. Both approaches have the disadvantage of an expensive treatment of the error function and their derivatives and are restricted to second-order regularization, at least for this simple formulation. The usage of algebraic kernels in the present work has no such drawbacks. Their implementation and calculation is simple and with the procedure presented in Theorem 3.1.3 new kernels with arbitrary order can be derived easily.

It is worth mentioning that a first step towards a multipole expansion theory with algebraic kernels has been presented in [100]. The theory derived there is focused on the second order algebraic kernel and the error term is reproduced by our theory for $G_\sigma^{(2)}$. Moreover, they describe a generalization of a recurrence relation for the multipole expansion of this kernel, originally proposed in [101]. With the help of the Decomposition Theorem, this relation scheme can easily be extended to the classes S_σ^r, A_σ and $\bar{A}_\sigma$, which is not part of our work, though.

3.3 Summary

In this chapter, we have adapted the concept of multipole expansions for the case of regularized vortex particle methods using algebraic kernels. Figure 3.5 updates the introductory choice between cutoff-based and multipole-based methods in favor of the latter approach. The definition and analysis of multipole expansions for algebraic kernels of arbitrary order was the topic of this chapter.

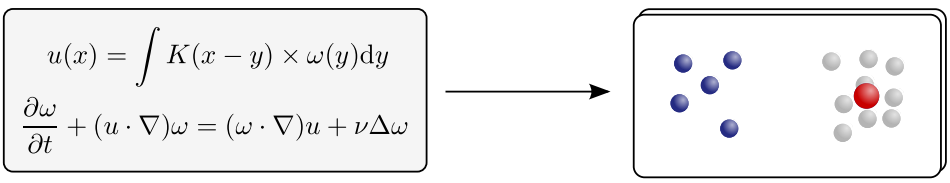


Figure 3.5: The transition from analytic equations to multipole-based methods.

We have derived a new and comprehensive class A_σ of algebraic smoothing kernels, comprising and extending the well-known zeroth and second order algebraic kernels. We have shown that within the framework of this class, kernels of arbitrary order can be derived easily. An elegant decomposition into members of a much simpler class D_σ was the basis for an extensive analysis of A_σ and its members. Especially, this decomposition eased the access to multipole expansion theory radically. We derived analytical error bounds and deduced convergence conditions for arbitrary members of both classes, which turned out to nicely extend well-known results for ρ^{-n} kernels to their regularized counterparts. This theory now yields a reliable and mathematically well-grounded basis to apply multipole-based tree codes to N-body problems arising from vortex particle methods with algebraic kernels of arbitrary order.

In terms of Algorithm 2.1, we are now able to replace the expensive and dominating subroutine COMPUTEINTERACTION with a call to TREECODE. With this modification we are able to reduce the complexity from $\mathcal{O}(N^2)$ to at least $\mathcal{O}(N \log N)$, making simulations with millions of particles economically reasonable.

In the next chapter, we will address the implementation, analysis and application of one of these multipole-based algorithms, namely the parallel Barnes-Hut tree code PEPC [102]. Although the step from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$ is already an important improvement, a careful parallelization can increase usability and speed of the algorithm even further. Besides an overview of the workflow, we will therefore highlight the parallelization strategy and its impact on performance and capabilities as well.

4 Implementation of a parallel tree code for vortex dynamics

In the previous chapter, we have formulated the mathematical basis for multipole-based methods with algebraic kernels. The most important property of this theory for an implementation is the separation of multipole moments and multipole expansion as stated in Definition 3.2.1. In multipole-based algorithms, the simulation region is typically decomposed into a hierarchical oct-tree structure. This data structure allows for an efficient a priori computation of the multipole moments. Starting from particle-level, the multipole moments are shifted down the tree to root-level, equipping each node with moments up to a given order p. The tree is then traversed to identify interacting particles and nodes, i.e. groups of distant particles. Figure 4.1 illustrates two related implementations: the Fast Multipole Method (FMM) [83] and the Barnes-Hut tree code [82]. Besides the mathematical motivation, both methods share the octal decomposition scheme of the simulation region, i.e. the way of structuring the particles into boxes. Furthermore, both methods implement the precomputation of the multipole moments in a similar way, so that the global construction of the tree-like data structure can be seen as the common feature of both approaches. However, they differ in some fundamental aspects, which we highlight in the following.

Barnes-Hut tree code – The Barnes-Hut tree code explicitly makes use of the multipole expansion as explained in Section 3.2. After the construction of the tree structure with fixed,

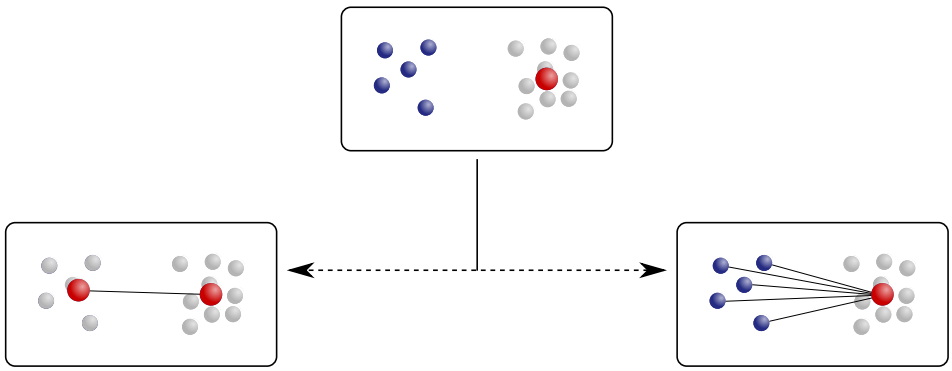


Figure 4.1: Fast summation using FMM or Barnes-Hut tree codes?

predefined expansion order p, the multipole acceptance criterion (MAC) can be used to systematically traverse the tree and to identify the interacting particles or nodes. However, this process is always performed for a single particle and not for a whole group of particles. For each particle the tree is traversed, starting from root-level, until for one node the MAC is satisfied. Therefore, one of the interacting partners always is a particle, motivating the term 'particle-box-interaction'. This concept leads to an $\mathcal{O}(N \log N)$-scheme [90, 103].

FMM – While the Barnes-Hut tree code can be seen as a direct implementation of the theory of multipole expansions of Section 3.2, the FMM exploits the grouping of particles even further. Here, so-called 'box-box-interaction' are explicitly allowed and required. This implies that not only operators for the down-shift of multipole moments are necessary, but also operators for up-shifting as well as expansion transformations. These operators often rely on a spherical expansion of the kernel, in contrast to the Cartesian approach of Barnes-Hut tree codes. Furthermore, the expansion order p is commonly kept variable to allow for a rigorous error-control scheme [104, 105]. The consistent use of box-box-interactions yields an $\mathcal{O}(N)$-scheme, thus minimizing the practical complexity of N-body simulations [91].

Multipole-based methods have become the standard 'mesh-free' means for long-range N-body simulations across a broad range of fields [89]. Their sophisticated mathematical background allows for rigorous error estimates and convergence analyses. In contrast to the choice of Chapter 3 – between multipole-based and cutoff-based methods – the present decision is motivated by design aspects and not by a clear weighing of pros and cons. The theory developed in this work is valid and applicable for generic multipole-based approaches in the context of vortex particle methods with algebraic smoothing kernels. However, we have left out the derivation of the two additional FMM-operators, which would exceed the scope of this work. To a certain degree, the Barnes-Hut tree code approach can be interpreted as a reduced FMM. This reduction comes at the costs of an additional $\log N$-term in the theoretical complexity, but on the other hand the difficulty of parallelization is somewhat lessened.

With an effective parallelization strategy at hand, a multipole-based method can fully exploit the advantages of distributed-memory architectures, thus rendering a direct $\mathcal{O}(N^2)$-scheme dispensable. Therefore, the goal of this chapter is a comprehensive description of one parallel Barnes-Hut tree code for vortex particle methods, its capability and its application. The code presented in this work bases on the parallel tree code PEPC – the Pretty Efficient Parallel Coulomb-solver [102, 106, 107, 108] – using its parallelization strategy and data management. In the following, the expression 'tree code' refers to the Barnes-Hut tree code, unless stated otherwise.

In the first section, we describe the workflow of the parallel tree code PEPC with its adaptation to vortex particle methods. The implementation described here has been rewritten completely to match the requirements of regularized vortex particles. This includes a novel approach for remeshing by parallel sorting, making this process a generic extension of tree codes. We briefly analyze the theoretical parallel complexity of the algorithm and show scaling results

in N and P for a spherical vortex sheet. Furthermore, we test our code using a vortex ring simulation and compare it with other vortex particle implementations.

4.1 The workflow of the parallel tree code PEPC

In [107, 108] we demonstrated the performance and scalability of the parallel tree code PEPC – the Pretty Efficient Parallel Coulomb-solver – on the IBM Blue Gene/P system JUGENE at Jülich Supercomputing Centre. This highly portable code was designed primarily for mesh-free modelling of nonlinear, complex plasma systems [84, 109]. Based on the original Warren-Salmon 'hashed oct-tree' scheme [110, 111] with a multipole expansion up to $p = 2$, PEPC provides a flexible and fully parallelized framework for $\mathcal{O}(N \log N)$ tree code simulations [106]. This framework is the basis for our adaptation to vortex particle methods.

The basic workflow of the tree code is depicted in Algorithm 4.1. Its concept is as follows: In the domain decomposition procedure, we generate octal keys out of the coordinates and sort them in parallel. According to this sorted list we subsequently redistribute the input particle set across the available tasks, obtaining a truly data-parallel scheme. Each process with its new particle set can now allocate a tree structure, primarily composed of a hash table and appropriate arrays for the tree properties. After the tree construction step the keys are structured as an oct tree with multipole moments attached to each node. Now, due to possible memory restrictions we subdivide the local particle set into chunks: For each particle inside these chunks the tree traversal fills up the local tree with required non-local information. For each pair of particle and node, the interaction are directly performed according to the multipole expansion formula if the multipole acceptance criterion is satisfied. Here, a node can be single particle or a cluster of particles.

Algorithm 4.1 Workflow of the parallel tree code

```
subroutine TREECODE                          ▷ to be called from an application
    call DOMAINDECOMPOSITION
    call TREECONSTRUCTION
    decompose local particle set into chunks 1 to K
    for k = 1 to K do
        call TREETRAVERSAL(k)
    end for
end subroutine
```

In the following, we review the three essential steps

- step 1: domain decomposition,
- step 2: tree construction,
- step 3: tree traversal,

of the algorithm in more detail and point out the difficulties arising from parallelization. Afterwards, we will show one efficient way of incorporating remeshing into a parallel tree code.

4.1.1 Domain decomposition

The initial step is the generation of 64-bit keys out of the coordinate triple of each particle inside a simulation area $\mathcal{A}$. These keys provide a simple yet powerful basis for sorting, tree construction and traversal. They are constructed from the 'binary interleave operation'

$$\texttt{key} = \texttt{placebit} + \sum_{j=0}^{\texttt{nlev}} 8^j (4 \cdot \text{IBIT}(n_3, j, 1) + 2 \cdot \text{IBIT}(n_2, j, 1) + 1 \cdot \text{IBIT}(n_1, j, 1)),$$

where the function $\text{IBIT}(a, b, l)$ extracts l bits of the integer a starting from position b. The components of the normalized coordinate vector (n_1, n_2, n_3) are defined as

$$n_k = (x_k - \min\{y_k : y \in \mathcal{A}\}) \cdot \frac{2^{\texttt{nlev}}}{L},\ k = 1, 2, 3$$

with simulation box length L and the maximum refinement level `nlev`. With a 64-bit unsigned integer data type for the keys, 21 bits per coordinate (which is $\texttt{nlev} = 20$) and an additional place-holder bit

$$\texttt{placebit} = 2^{3(\texttt{nlev}+1)} = 2^{63}$$

are available. The necessity for such a place-holder bit becomes clear in the tree construction step and is explained in Section 4.1.2. The restriction of 21 bit per coordinate leads to a resolution bound of approximately $2^{-(\texttt{nlev}+1)} \approx 5 \times 10^{-7}$ within the unit box. Below this threshold two distinct particles would map onto the same key. This mapping of coordinates to keys yields a Morton- or Z-ordering, which is a special case of general space-filling curves [112]. In Figures 4.2 and 4.3, we give a short example for the Z-ordering and redistribution with 52 particles and 3 tasks in two spatial dimensions. After the key construction the particles are sorted and redistributed among the tasks according to the list of keys. The sorting routine is fully parallelized and contains the key load-balancing features of the algorithm. During the simulation each particle is tagged with a work load quantity depending on the force calculation costs. The total aggregated work load on each task controls its allocation of particles in the following time step. This procedure is crucial: an imbalanced work load may significantly slow down the simulation. More details on different parallel sorting approaches in the context of PEPC can be found in [107, 113].

After the domain decomposition each task is equipped with a set of local particles, which are disjoint except for the particles at task boundaries: here, we copy these edge particles to the next task to avoid problems with leaf sharing throughout these boundaries.

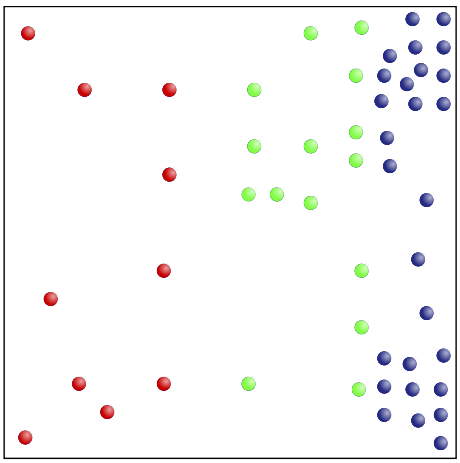


Figure 4.2: Sample initial 2D configuration with 52 particles and 3 tasks.

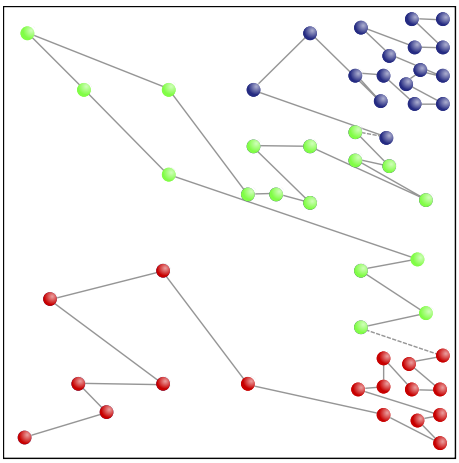


Figure 4.3: Z-ordering and redistribution of the 52 particles on 2 tasks.

4.1.2 Tree construction

The reason for the octal key construction procedure is its usability in a parallel tree data structure. Keys of parent or child cells can be obtained by simple bit operations that speed up the access-time within the tree structure significantly. Simple 3-bit shift operations of the form

$$\text{GETPARENT} = \text{BSHIFT}(\texttt{key}, -3)$$

derive the parent from the actual tree node. For obtaining the key of one of the potentially eight children of a cell we store for each parent the number and position of its children in a `childbyte` variable, which can subsequently be tested with IOR – the boolean OR of pair of bits. Using BTEST(`childbyte`,`id`) we can check, whether or not the `id`-th child is present. The operation

$$\text{GETCHILD}(\text{id}) = \text{BSHIFT}(\texttt{key}, 3) \text{ IOR } \texttt{id}.$$

then returns the corresponding key. As a simple two-dimensional example, in Figure 4.4 the full key for the highlighted cell is 131 in quadric, 11101 in binary notation. We have dropped the place-holder bit in the figure for clarity, but it is crucial for the implementation. Prepending an additional bit – the place-holder bit – to the most significant bit of every key allows us to represent all higher level nodes in the tree in the same key space [111]. Without this scheme the key 0001 would have an undefined level between 0 and 3 since the leading zeroes could be part of the key itself or just undefined bits.

Nevertheless, the general key construction approach has an inherent drawback. The number of possible keys vastly exceeds the available memory, so that direct addressing is not feasible

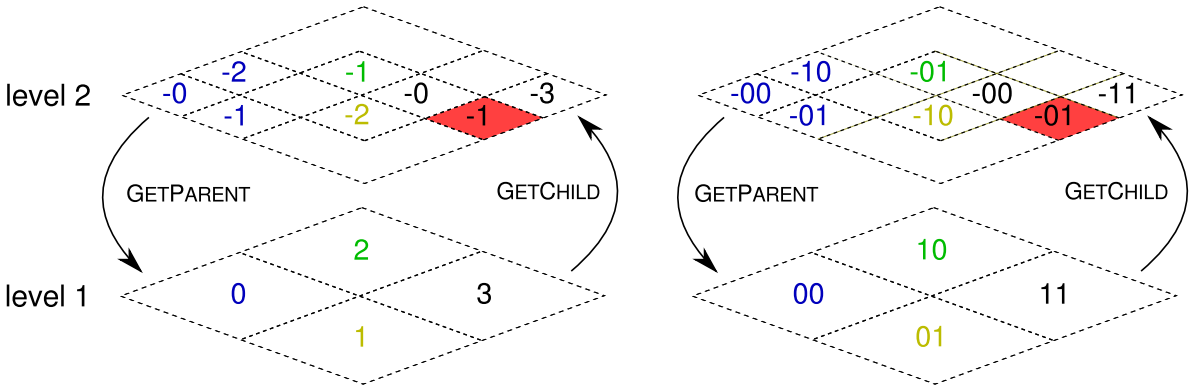


Figure 4.4: Obtaining parent and child key from node: quadric and binary notation.

and often not even possible. Therefore, a convenient way of condensing the required storage is to use a hashing function that maps the key space onto physical addresses in memory. We use a simple `AND` operation with

$$\texttt{address} = \texttt{key AND } (2^{\texttt{nb}} - 1),$$

where `nb` is the number of bits available for the address' space. This operation selects the least significant `nb` bits of the initial 64-bit keys. In case of a collision, i.e. a mapping of two or more keys to the same location, we use a linked list for resolving the keys. It is clear, that a disadvantageous choice of a hash function affects the algorithm's performance significantly due to frequent collisions. As noted in [111], the simple `AND` operation performs extraordinarily well in this matter, which gives us a convenient yet powerful way of structuring our data.

As a consequence, the underlying data structure for the tree consists of a hash table with entries containing

`node:`	hashed address of the corresponding node,
`key:`	its original key,
`link:`	a pointer on the next non-empty address in case of collisions,
`leaves:`	the number of leaves in the subtree,
`childcode:`	bit-encoded children number and positions,
`next:`	a pointer on the next sibling's, uncle's, great-uncle's, ... key,
`owner:`	the node's owner, i.e. the corresponding task id.

After its allocation and setup, this hash table contains all necessary data on each task for a tree traversal. To setup the hash table and thereby the tree structure, basically two steps are necessary: the local tree construction and the definition of global branch nodes.

a) For the local tree construction process, all local particles are initially attached to the root level covering the whole simulation area $\mathcal{A}$. Next, this region is subdivided into

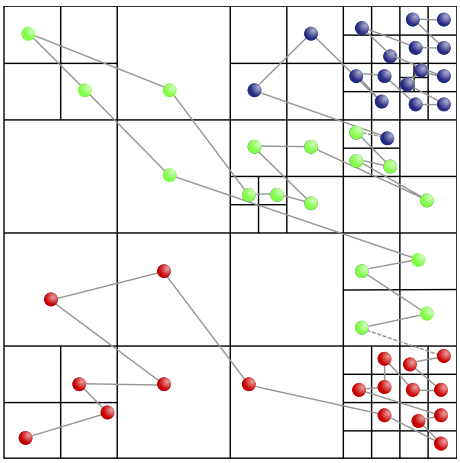


Figure 4.5: Octal partitioning of the simulation domain and allocation of 52 particles to a box, 3 tasks.

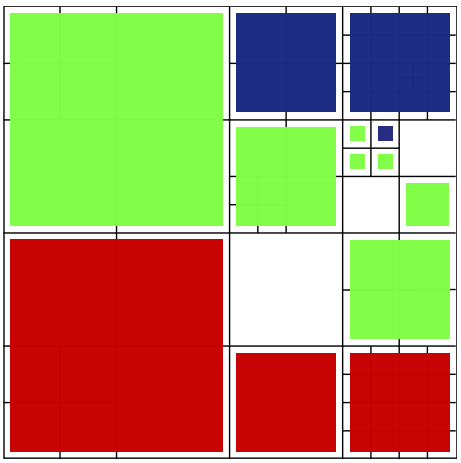


Figure 4.6: 'Branch' nodes covering the simulation domain of each task, 52 particles on 3 tasks.

eight sub-cells (four in two dimensions). This is recursively repeated for each sub-cell until each cell contains exactly one or no particles. A cell with one particle is defined as a 'leaf', a cell with two or more objects constitutes a 'twig', while empty cells are discarded. The creation of new leaves or twigs is done by adding the corresponding node information to the hash table using the hash function. At the end of this substep each task has its own local tree with incomplete twig nodes at the coarsest level, whose children partially belong to other tasks. Figure 4.5 continues our previous example with 52 particles on 3 tasks.

b) To complete the twig nodes on the coarsest level, information from neighboring domains is necessary. Therefore, each task defines its local set of 'branch' nodes, which comprise the minimum number of complete twig or leaf nodes covering its entire domain. The collectivity of all branch nodes represents a decomposition of the whole simulation area $\mathcal{A}$ at the coarsest level possible. After propagating these nodes to the other tasks, the branch nodes act as an entry point to non-local trees, so that subsequently every task is capable of requesting remote information during a tree traversal. Figure 4.6 highlights the branch level of each process within the context of our example.

It should be mentioned here that the global branch concept is prone to serious parallelization issues. On the one hand, the overall number of branches on each task is of order $\mathcal{O}(P)$ for P tasks. On the other hand, the ensemble of branch nodes has to be broadcasted across all tasks, which induces a potential parallelization bottleneck. We return to this issue in Section 4.2.2.

Finally, after the branch structure is set up and made globally available, the remaining tree structure on each task from the branches to the root level is filled up, so that a traversal may

start on each task at root node. Nodes between branch and root level are called 'fill' nodes. Concurrently, as we create leaf, branch and fill nodes, we calculate the multipole moments for every node in the tree. This is a simple operation for the monopole moments, because they are independent from the current cell's multipole origin or center-of-charge $\hat{x}$. But we are also dealing with dipole and quadrupole moments, which are inherently linked to the multipole origin. To avoid the summation over all nodes in the sub-tree, we have to exploit our tree structure. One major advantage of this structure is that the difference of $\hat{x}$ from one node to its parent can be expressed by a simple shift operation [89].

To clarify this shifting procedure, we analyze the multipole moments

$$M_d^{(0,0,0)},\ M_d^{(1,0,0)},\ M_d^{(2,0,0)},\ M_d^{(1,1,0)}$$

for given child-cells $\mathcal{V}_d$ with $d = 0, \ldots 7$, adopting our previous notation of Definition 3.2.1

$$M_d^{(\beta)} = \sum_{y \in \mathcal{V}_d} (y - \hat{x}_{\mathcal{V}_d})^\beta \cdot (\omega(y)\mathrm{vol}(y)) = \sum_{y \in \mathcal{V}_d} (y - \hat{x}_{\mathcal{V}_d})^\beta \cdot \alpha_y.$$

We define their corresponding multipole origins by

$$\hat{x}_{\mathcal{V}_d} = \frac{\sum_{y \in \mathcal{V}_d} |\alpha_y| \cdot y}{\sum_{y \in \mathcal{V}_d} |\alpha_y|}.$$

If $\hat{x}_{\mathcal{W}}$ denotes the origin of the parent cell $\mathcal{W}$ of $\mathcal{V}_d$, the shift vector is simply

$$\tilde{x}^d = (\tilde{x}_1^d, \tilde{x}_2^d, \tilde{x}_3^d)^T = \hat{x}_{\mathcal{W}} - \hat{x}_{\mathcal{V}_d}.$$

Now, the multipole moments expressed in terms of the new origin $\hat{x}_{\mathcal{W}}$, are

$$\begin{aligned}
\tilde{M}_d^{(0,0,0)} &= M_d^{(0,0,0)} = \sum_{y \in \mathcal{V}_d} \alpha_y, \\
\tilde{M}_d^{(1,0,0)} &= \sum_{y \in \mathcal{V}_d} (y - \tilde{x}^d)_1 \cdot \alpha_y = M_d^{(1,0,0)} - \tilde{x}_1^d M_d^{(0,0,0)}, \\
\tilde{M}_d^{(2,0,0)} &= \sum_{y \in \mathcal{V}_d} (y - \tilde{x}^d)_1^2 \cdot \alpha_y = M_d^{(2,0,0)} - 2\tilde{x}^d M_d^{(1,0,0)} + (\tilde{x}_1^d)^2 M_d^{(0,0,0)}, \\
\tilde{M}_d^{(1,1,0)} &= \sum_{y \in \mathcal{V}_d} (y - \tilde{x}^d)_1 (y - \tilde{x}^d)_2 \cdot \alpha_y \\
&= M_d^{(2,0,0)} - \tilde{x}_1^d M_d^{(0,1,0)} - \tilde{x}_2^d M_d^{(1,0,0)} + \tilde{x}_1^d \tilde{x}_2^d M_d^{(0,0,0)}.
\end{aligned}$$

Therefore, all moments $M^{(\beta)}$ of the parent cell $\mathcal{W}$ can be obtained from the moments of the child-cells $\mathcal{V}_d$ and the shift vector $\tilde{x}^d$. We get

$$\begin{aligned}
M^{(0,0,0)} &= \sum_{d=0}^{7} \tilde{M}_d^{(0,0,0)}, \quad M^{(1,0,0)} = \sum_{d=0}^{7} \tilde{M}_d^{(1,0,0)}, \\
M^{(2,0,0)} &= \sum_{d=0}^{7} \tilde{M}_d^{(2,0,0)}, \quad M^{(1,1,0)} = \sum_{d=0}^{7} \tilde{M}_d^{(1,1,0)}.
\end{aligned}$$

Since each task is now equipped with a complete tree structure and accumulated multipole moments on the nodes, the next step is the traversal of the tree for each particle to gather its interacting nodes and to compute the multipole expansion.

4.1.3 Tree traversal

The tree traversal routine is the most important and communication intensive part of our algorithm. The aim of this procedure is the assembly of the 'locally essential tree' on each process, which contains all necessary information for the multipole expansion. Here, we use an asynchronous requesting scheme for gathering missing non-local node properties similar to that in [111]. Rather than performing one traversal for one particle at a time, we reduce the algorithm's overhead by initializing as many traversals as possible, bounded only by machine-dependent memory constraints. A simultaneous tree traversal for a predefined set of active particles aims at minimizing the amount of duplicated request while maximizing the available communication bandwidth by requesting and fetching a reasonable large number of nodes per step [110].

During the traversal the multipole acceptance criterion (MAC) is responsible for the decision whether or not a node has to be resolved further. This test can be based on pure geometric considerations or even include information from the nodes. Its choice influences both speed and accuracy of a tree algorithm and has to be decided carefully. We have already introduced two possible choices in Section 3.2.4:

a) Motivated by the convergence analysis of multipole expansions, a common choice in tree codes is the 'Bmax MAC', which defines a node covering a cluster $\mathcal{V}$ as sufficient if its maximal extend b is related to the distance between the current particle x and the center-of-charge $\hat{x}$ by

$$\theta > \frac{b}{d} = \frac{\sup\{|y - \hat{x}| : y \in \mathcal{V}\}}{|x - \hat{x}|}$$

with a predefined and fixed θ. Choosing $\theta = 1$ already yields a convergent yet inaccurate scheme, while $\theta = 0$ implies an exact $\mathcal{O}(N^2)$-algorithm. A convenient compromise between accuracy and speed is $\theta \in (0.4, 0.7)$.

b) The expression for the error term

$$|\varepsilon^{\tau}_{(p+1)}(x)| \leq \mathcal{D}_{2\tau-1}(p+1) \cdot \frac{\bar{M}_{(0)}}{(d-b)^{2\tau}} \cdot \left(\frac{b}{d-b}\right)^{p+1} \leq \Delta\varepsilon$$

can be used to define a data-dependent MAC with a predefined threshold $\Delta\varepsilon$. A node is acceptable only if

$$d \geq b + \left(\frac{\mathcal{D}_{2\tau-1}(p+1) \cdot \bar{M}_{(0)} \cdot b^{p+1}}{\Delta\varepsilon}\right)^{\frac{1}{p+1+2\tau}}.$$

A detailed discussion of geometric and data-dependent criteria with a rigorous analysis can be found in [95].

Algorithm 4.2 gives an overview of the full traversal routine. For each active particle with an index between p_k to p_{k+1} of the current chunk k the tasks start the traversal one level below the root, following the global nodes into their own local tree or ending at a branch node. If the MAC is not satisfied and if the node's children are absent, we put this node onto a request list and mark the current traversal for later processing. The functions FIRSTCHILD and NEXTNODE use the `next`-attribute, which is part of each node's hash entry. If the MAC is satisfied for one pair of particle and node we sum up the contribution of the node directly using the routine CONTRIBUTION. Since our tree code uses expansions up to order two, i.e. quadrupole order, we compute with Definition 3.2.1

$$\phi_{(\beta)}(x) = \frac{(-1)^{|\beta|}}{\beta!} M^{(\beta)} \cdot D^{\beta} G_{\sigma}(z)\big|_{z=x-\hat{x}},$$

for $\beta \in \{(0,0,0),(1,0,0),\ldots,(0,0,2)\}$. Here, the precomputed and shifted multipole moments $M^{(\beta)}$ of Section 4.1.2 become crucial. They can be directly obtained from the tree structure without further calculations, while the derivative of $G_{\sigma}(z)$ requires the active particle x and the current center-of-charge $\hat{x}$. The latter value is again extracted out of the tree structure.

At the end of the first part, each task can provide request lists and parts of the locally essential tree for every active particle. In the following step, we exchange the request lists and initiate non-blocking send and receive operations, before every task can deal with incoming requests and returning child information. The returning data is sorted into the local tree structure for further traversal. After finishing all active traversals, i.e. as soon as all traversals terminate at nodes which satisfy the MAC, we finally end up with the locally essential tree for the current set of active particles and have access to their interacting nodes.

Furthermore, we define the work load per particle by the number of its interaction partners, assuming that the average costs for the identification of these nodes are constant over tasks. The sum over all local particle work loads yields the work load per task, which is the basis for a balanced domain decomposition in the following time step.

To conclude this description of the implementation, Figure 4.7 shows a simple 1D example of a serial and parallel tree data structure [107]. The top view in this figure shows nodes and particles as they would be organized in a sequential version of the code. Each particle is identified as a leaf `L`, while all other nodes are globally available fill nodes `F`. A traversal starts from root level, checks the multipole acceptance criterion and processes the whole structure up to the leaves, if necessary. The figure below depicts the parallel version of this structure. Leaf nodes are distributed across the tasks. For the traversal, the branch nodes `B` are necessary, acting as an entry node to non-local trees during the traversal. These nodes have to be available on every task in addition to their local trees and the global structure from root

Algorithm 4.2 Outline of the traversal routine

```
subroutine TREETRAVERSAL(k)
    node(p_k ... p_{k+1}) ← FIRSTCHILD(root)
    force(p_k ... p_{k+1}) ← 0

    while traversals active do

        while particles active do
            p ← active particle
            if MAC(p,node(p)) ok then
                force(p) ← force(p) + CONTRIBUTION(node(p))
                node(p) ← NEXTNODE(node(p))
            else if MAC failed and node(p) is local then
                node(p) ← FIRSTCHILD(node(p))
            else if MAC failed and node(p) is non-local then
                put node(p) on request list, discard duplicates
                mark node(p) as active traversal
                node(p) ← NEXTNODE(node(p))
            end if
        end while
        mark finished traversals as inactive

        while requests open do
            test for incoming requests
            package and ship back child multipole info
        end while

        while receives open do
            test for returning child multipole info
            create hash entries for each child
        end while

    end while
end subroutine
```

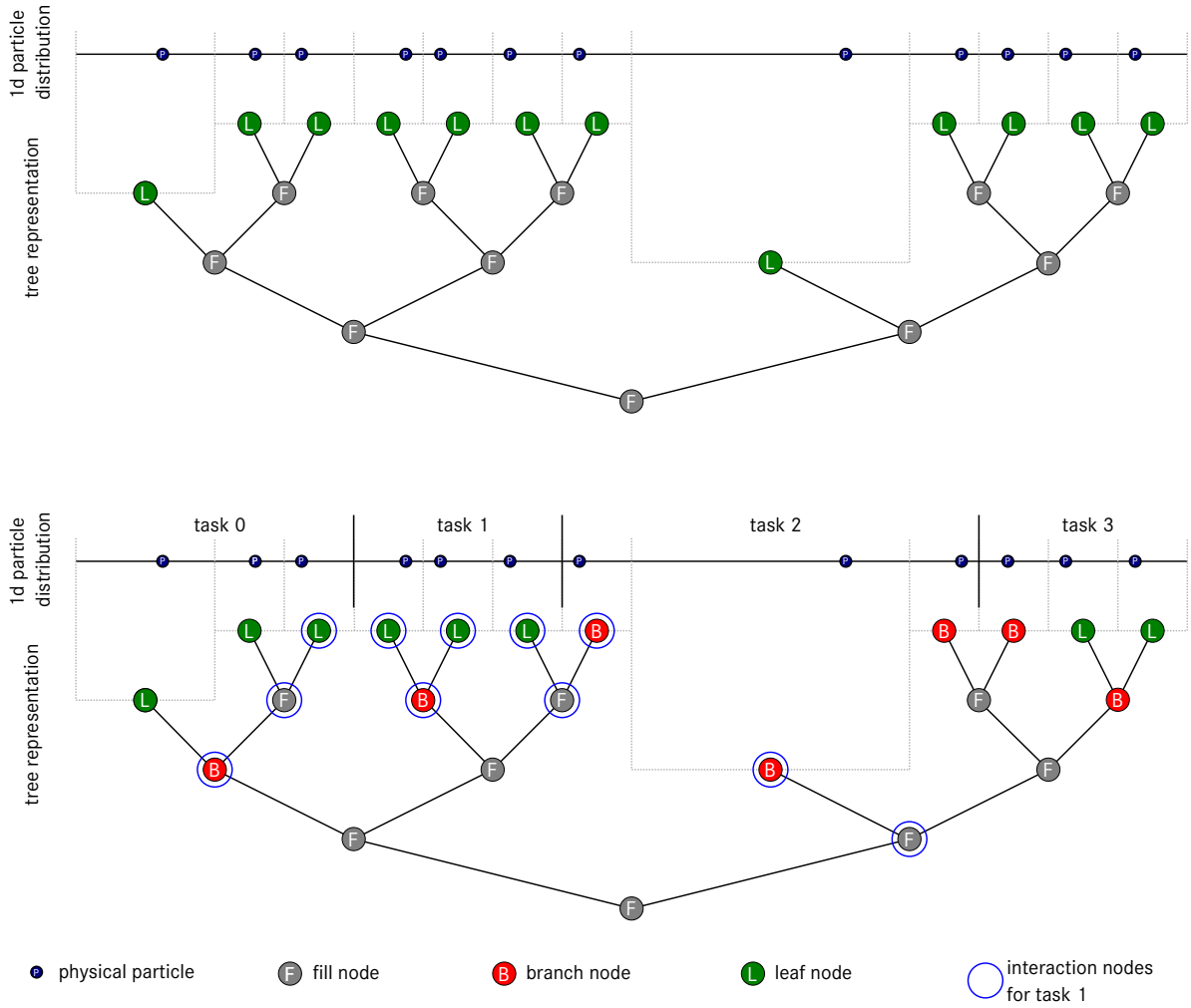


Figure 4.7: 1D example of the tree structure. Top: particle distribution with the according leaf nodes `L` and the corresponding fill nodes `F` as they would be organized in a serial version of the code. Bottom: leaf nodes are distributed across the tasks but for a consistent data structure the branch nodes `B` are necessary and act as an entrance node to non-local trees during tree traversal. Furthermore, we have indicated the interacting nodes with particles on task 1.

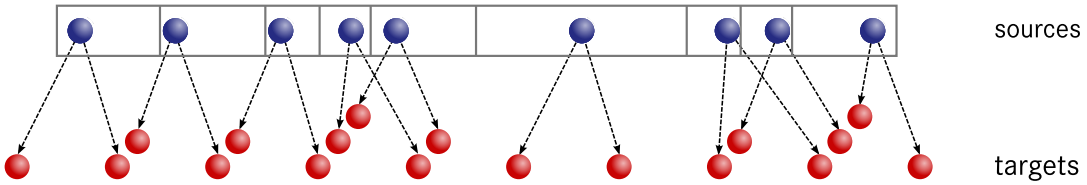
to the branches. Furthermore, we have indicated nodes that interact with particles on task 1. Since only these nodes are necessary for the multipole expansion, some parts of the branch and global structure stored in the tree of task 1 turn out to be redundant. We will discuss the impact of this effect in Section 4.2.2.

4.1.4 Parallel remeshing

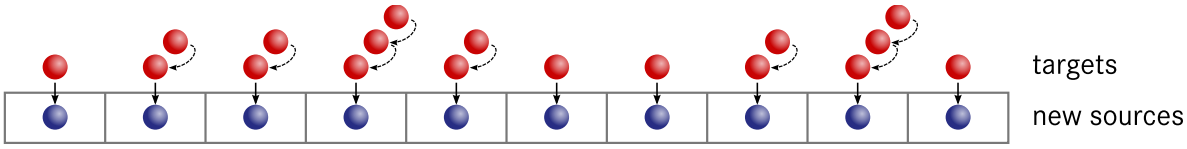
Although not part of the standard tree routines, an optional remeshing procedure according to Section 2.2.2 can be integrated into an existing parallel tree code, maintaining the overall complexity of $\mathcal{O}(N \log N)$. Without an underlying mesh structure, an implementation of location processing techniques has the inherent problem of finding the interpolation nodes. Since the original particles are not linked to each other, the generation of $M = \mathcal{O}(N)$ new particles would require $\mathcal{O}(N^2)$ operations. Many mesh particles may have more than one source and it is necessary to identify these conflicting projections. This problem is a question of finding duplicates within a given set of values without testing the whole set.

One solution may involve the use of a global mesh data structure, where each source particle can pick its targets. Although easy to implement, many parts of this structure may remain empty, since no source activates any targets there. This would completely compromise the mesh-free character of vortex particle methods. The usage of sparse data structures like a hash table can circumvent this drawback, but then a parallelized remeshing scheme has to be considered as complex as the original tree code. Local hash tables are required to maintain data parallelism and the identification of duplicate targets across task boundaries involves non-trivial and expensive operations. Alternatively, the original tree data structure can be used. As presented in [55], target particles are created above current leaves, i.e. source particles, and their strengths are determined from particles nearby by traversing the tree. This reduces the memory requirement significantly, since the set of target particles is minimal. Nevertheless, one has to provide a priori storage for the targets, so that for a remeshing scheme using k neighbors in each of the three spatial dimensions, $N \cdot k^3$ entries have to be reserved for the targets. This drawback is common for mesh-free strategies, since duplicates cannot be identified a priori in these cases. However, the number of entries has a clearly defined upper bound and these techniques preserve the intrinsic adaptivity of particle-based methods. Both properties are not valid for mesh- or matrix-based structures as described above.

We therefore choose a purely particle-based approach for our remeshing implementation. However, the approach presented in [55] requires frequent and expensive tree traversals for each target particle to identify its source particles. Our novel approach avoids these operations by exploiting the parallel sorting algorithm within the domain decomposition. The basic idea is depicted in Figure 4.8 for a simple one-dimensional example. In 4.8a, nine source particle are inhomogeneously distributed and need to be interpolated onto a homogeneous set of target particles, using only two nearest targets for simplicity. To this end, we create for each source a local set of its targets. Conflicting projections are indicated by diagonally arranged target particles, yielding $9 \cdot 2^1 = 18$ target particles. To identify and combine the duplicates, we generate particle keys for each target and use the parallel sorting algorithm. This redistributes the targets so that duplicates are located one after another. A simple sweep over all targets as depicted in 4.8b then combines these duplicates to one single particle by accumulating the



(a) Interpolation from source particles to local targets



(b) Aggregation of doublets and transformation to new source particles

Figure 4.8: Nine source particles are interpolated onto new particles located at nodes of a uniform mesh. For clarity, the interpolation uses only the two nearest mesh particles as targets. Diagonal particles indicate doublets. These are combined by parallel sorting to a single new source particle in the second step.

particular strengths and by erasing obsolete target particles. Finally, we define the unique targets as new source particles.

The advantages of this approach are its simplicity and efficiency: an existing domain decomposition algorithm with parallel sorting can be easily extended by parallel remeshing without affecting its own theoretical complexity. Since each source particle uses only a fixed number of targets, defined by the support of the interpolation operator, the complexity remains the same. Furthermore, this approach yields again the minimum number of active mesh particles without sustaining a full mesh data structure. No expensive tree traversals are required and we can separate the remeshing process from the actual tree construction, thus avoiding needless entries for the obsolete source particles. Moreover, the parallelization is straightforward if we are already equipped with a parallelized sorting algorithm. The identification of duplicates across task boundaries only requires one additional array of keys, where each task inserts its first and last key. After broadcasting this array, each task can check for duplicates itself, even if in rare cases these duplicates are spread across more than one task.

However, the drawback of our approach is again the possible storage overhead due to the generation of a vast amount of duplicates. But since an a priori identification of obsolete targets is not possible for purely particle-based techniques, the storage has to be provided anyway, as mentioned above.

We therefore conclude that even without an underlying mesh data structure, the remeshing approach can be integrated into the domain decomposition of a parallel tree code, maintaining efficiently the mesh-free character and complexity of the algorithm.

4.2 Capability analysis and challenges

The adaptation of an algorithm to massively parallel systems with thousands of tasks prompts a multitude of challenges. In general, these differ from those which arise at scales of only a few tasks. In the case of strong scaling at large scales, all activities that have a constant or even a P-dependent complexity must be avoided. However, some patterns might contain essential non-scaling actions, like collective communication, which have to be optimized or completely redesigned.

In the first part of this section, we briefly analyze the theoretical parallel runtime complexity of the tree code. For the upcoming scaling tests in N and P we introduce the setup of a spherical vortex sheet, a common test for three-dimensional vortex methods. By the means of this setup, we compare the scaling in N to a direct $\mathcal{O}(N^2)$-algorithm and demonstrate parallel scaling on a massively parallel system, the IBM Blue Gene/P JUGENE at Jülich Supercomputing Centre. Since PEPC targets strong scaling, there is unavoidably a maximal number of tasks up to which a speedup can be gained, due to essential non-scaling elements. In order to see how these bottlenecks might be minimized, the following sections also give an insight in the efficiency of the communication strategy, as well as the identification of one of the current bottlenecks: the P-dependency of the number of branch nodes.

4.2.1 Parallel runtime complexity

We have seen in the previous sections that even with active remeshing the code can be divided into three main parts:

- step 1: domain decomposition and remeshing,
- step 2: tree construction,
- step 3: tree traversal.

For a serial tree code and a homogeneous particle distribution, the theoretical runtime complexity is of order $\mathcal{O}(N \log N)$ for N particles [89, 90, 103]. However, in very special worst cases the complexity can grow up to $\mathcal{O}(N^2)$. Figure 4.9 shows a setup, in which a new level in the tree has to be created for each particle. For an algorithm with arbitrary tree depth, this setup leads to computational costs of order $\mathcal{O}(N^2)$ for visiting each particle (see [105] for a similar setup in the context of an FMM implementation). Since this case is somewhat pathological, it suffices to observe that in practice, a serial tree code scales with $\mathcal{O}(N \log N)$. Further complexity statements shall be interpreted in this way.

Using P tasks instead of one not only requires a lot of additional implementation effort, as we have seen in the previous sections, but also introduces parallel overhead like the branch structure. The following paragraphs briefly analyze the impact for the three parts. We assume N homogeneously distributed particles and P equally equipped tasks and define $n_p := \frac{N}{P}$. Fur-

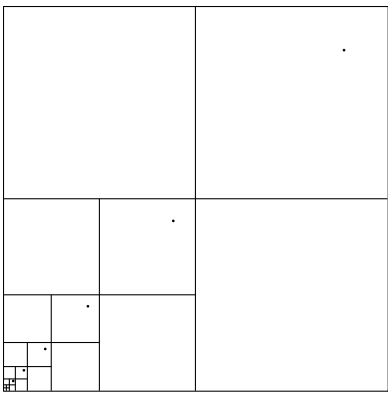


Figure 4.9: Worst case setup: one particle in each upper right box of each sub-box requires an additional level in the tree.

thermore, we omit the impact of Message Passing Interface (MPI) routines such as ALLREDUCE and ALLTOALL. We refer to [114] for a detailed analysis of these costs.

Domain decomposition – The decomposition step as well as remeshing is clearly dominated by the parallel sorting, since key construction and local interpolation require only $\mathcal{O}(N)$ operations. The sorting routine can be divided into a local sort and permute, global partitioning and redistribution and finally a local merge step. While local sort and permute are of order $\mathcal{O}(n_p)$ and scale very well, the global operations need $\mathcal{O}(n_p P)$ steps in the worst case, indicating potential bottlenecks for very large P. The final merge step is of order $\mathcal{O}(n_p \log P)$, so that the whole parallel sorting algorithm requires $\mathcal{O}(n_p + n_p P + n_p \log P)$ operations [113, 115].

Tree construction – As shown in [89], the construction of the tree with N particles induces costs of $\mathcal{O}(N \log N)$. During local tree construction, only n_p particles generate the initial tree, so that we can assume $\mathcal{O}(n_p \log n_p)$ operations here. Smart algorithms lead to negligible costs for the identification of the branch nodes, being the local roots of these trees. However, these branch nodes can then be seen as the leaves for the global tree construction step. Since we have $\mathcal{O}(P)$ branches, this step requires $\mathcal{O}(P \log P)$ operations. Therefore, the overall complexity for the tree construction is of order $\mathcal{O}(n_p \log n_p + P \log P)$.

Tree traversal – For homogeneously distributed particles, each particle requires $\mathcal{O}(\log N)$ interactions for the multipole expansion [103]. The exact number obviously depends on the MAC and its threshold θ, but as long as $\theta > 0$, the number of interactions scales like $\log N$ (see [89, 90, 103] for details). Since each task has to compute the interactions for n_p local particles, the total complexity of the tree traversal is of order $\mathcal{O}(n_p \log N)$.

In summary, the parallel tree code presented in Section 4.1 scales for N particles distributed over P tasks as

$$\mathcal{O}\left(\frac{N}{P}\left(\log N + P\right) + P \log P\right)$$

on average. The critical factor in the serial case – the N-dependency – can therefore be reduced with a parallelized version. However, this is not true for the $\log N$-dependency. In practice, the unfavorable P-term of the parallel sorting algorithm becomes visible only for very large values of P, potentially yielding a scaling saturation of the domain decompositions. On the other hand, the parallelization comes for the costs of an additional parallel overhead of order $\mathcal{O}(P \log P)$ – induced by the branch concept – which has a significant implication for the scaling behavior. In the next section, we analyze these results in a practical situation for vortex particle methods: the spherical vortex sheet.

4.2.2 Scaling in N and P

As a basis for the investigation of the scaling behavior, we choose the time evolution of a spherical vortex sheet. A given number of N particles is initially placed on a sphere with radius $R = 1$, centered at 0 (see [116] for details on homogeneous distributions on a sphere's surface) and attached with vorticity

$$\omega(x) = \omega(\rho, \theta, \varphi) = \frac{3}{8\pi} \sin(\theta) \cdot e_\varphi,$$

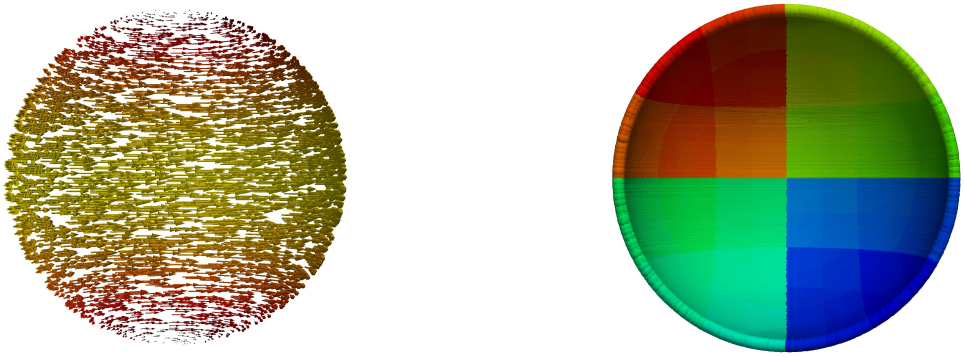
where e_φ is the φ-unit-vector in spherical coordinates. The volume or area of each particle is determined by the surface of the sphere, divided by N. Figure 4.10 visualizes this setup for $N = 10^5$ particles: (a) cuts the sphere along the (y, z)-plane and shows arrows for representing the vorticity in length and direction, (b) shows the distribution of the particles across the tasks after the first domain decomposition step with $P = 256$ tasks on the IBM Blue Gene/P system JUGENE.

For the dynamical simulation of this setup, the second order algebraic smoothing kernel

$$\zeta^{(2)}(\rho) = \frac{15}{8\pi(\rho^2 + 1)^{\frac{7}{2}}}$$

is used and expanded as described in Chapter 3. The time evolution of this setup can be seen in Figure 4.11. Neither remeshing nor diffusion are applied in these cases. As noted in [55], the initial conditions are the solution to the problem of flow past a sphere with unit free-stream velocity along the z-axis. While moving downwards in z-direction, the sphere collapses from the top and wraps into its own interior, forming a large vortex ring in the inside. This behavior is in very good agreement with [55, 99] and we take this setup as a basis for the scaling analysis in N and P.

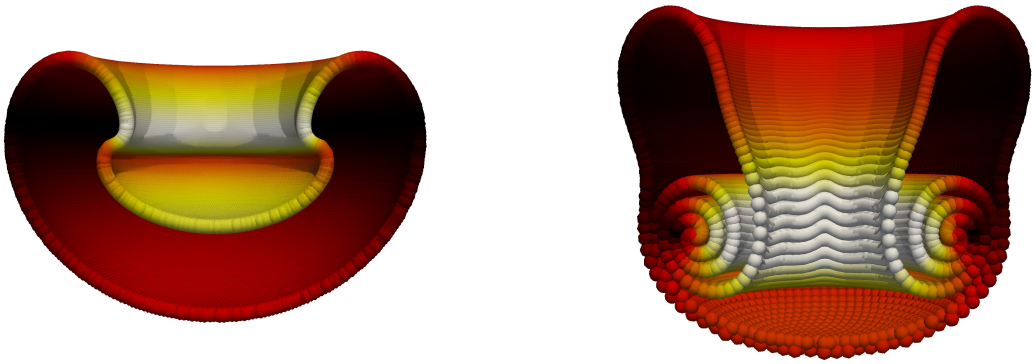
Before we analyze the scalability of the code for an increasing number of tasks, the first aspect we have to check is its scaling in N. To this end, we compare the setup for $N = 1000, \ldots, 32000$ particles on 2 MPI tasks. Figure 4.12 shows the execution times of the spherical vortex sheet, normalized to the case $N = 1000$. The runtime is an average over



(a) Cut along the (y, z)-plane

(b) Domain decomposition

Figure 4.10: Positions and vorticity of $N = 10^5$ particles on $P = 256$ tasks. In (a), the sphere is cut along the (y, z)-plane, for 2000 particles vorticity vectors are shown. Initial distribution across the tasks after the first domain decomposition step is depicted in (b).



(a) Time $t = 25.0$

(b) Time $t = 50.0$

Figure 4.11: Time evolution of the spherical vortex sheet at times $t = 25.0$ and $t = 50.0$, cut through (y, z)-plane. $N = 10^5$ particles, $P = 256$ tasks, Bmax-MAC with $\theta = 0.45$, $h = \sqrt{(4\pi)/N}$ and $\sigma = 0.22$, second order algebraic smoothing kernel. Third order Runge-Kutta time integration with $\Delta t = 0.05$. Light colors indicate larger velocity magnitudes.

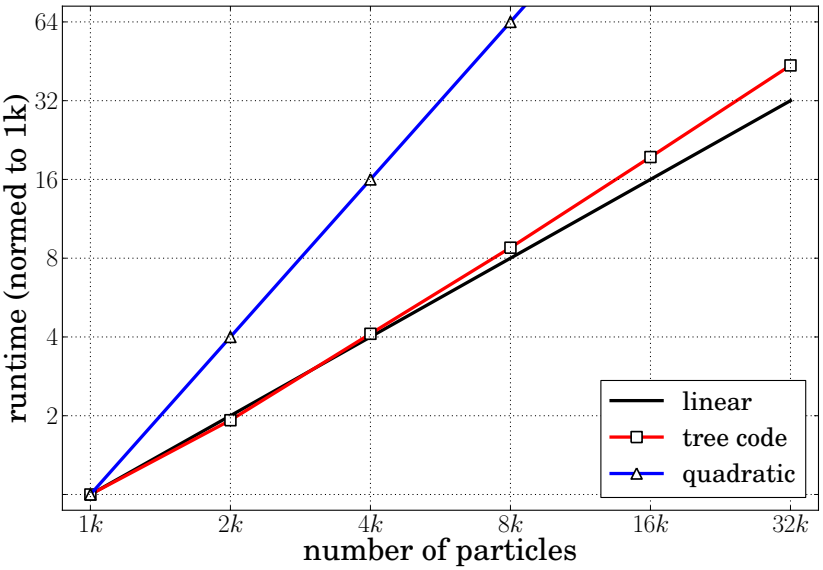


Figure 4.12: Normalized runtime for the simulation of a spherical vortex sheet with $N = 1000, \ldots, 32000$ particles on 2 tasks, average over 15 time steps (Third order Runge-Kutta time integration scheme with $\Delta = 1$). We choose $\sigma = 2h = 2\sqrt{(4\pi)/N}$. For comparison, theoretical linear and quadratic scalings are depicted. The tree code uses the Bmax-MAC with $\theta = 0.6$.

15 steps, using a third order Runge-Kutta time integration scheme with $\Delta t = 1$. We choose $\sigma = 2h = 2\sqrt{(4\pi)/N}$. The tree code uses the Bmax-MAC with $\theta = 0.6$. It becomes clear that the code scales nearly as well as the linear case. The $\log N$-term emerges for $N = 32000$ particles, but the practical complexity is far better than $\mathcal{O}(N^2)$, even for the inhomogeneous distributions during the simulation of the sphere.

We now analyze the scalability of the tree code for a fixed number of particles N, but increasing numbers of tasks P (strong scaling). Due to memory restrictions, we cannot expect to use the same number of particles for 1 to 8192 MPI tasks. That is why we divide our analysis into three sets: $N = 1 \cdot 10^5$, $N = 1.6 \cdot 10^6$ and $N = 2.56 \cdot 10^7$, i.e. $n_p = 10^5$ for 1, 16 and 256 tasks. We call the combined strong scaling test of these three setups 'hybrid scaling'. This analysis is related to a similar one for the purely Coulombian tree code PEPC-E [107, 108]. Figure 4.13 shows hybrid strong scaling results on JUGENE. As these runs show, our implementation is able to use up to 8192 MPI tasks very efficiently with adequate datasets. The hybrid scaling behavior is very good for 1 to 4096 MPI tasks, yielding more than three orders of magnitudes of usability and two orders of magnitude of speedup for one dataset. However, as the number of particles per task decreases, the scaling departs from ideal linear speedup. Furthermore, the appearance of this phenomenon is not only coupled to the ratio of particles

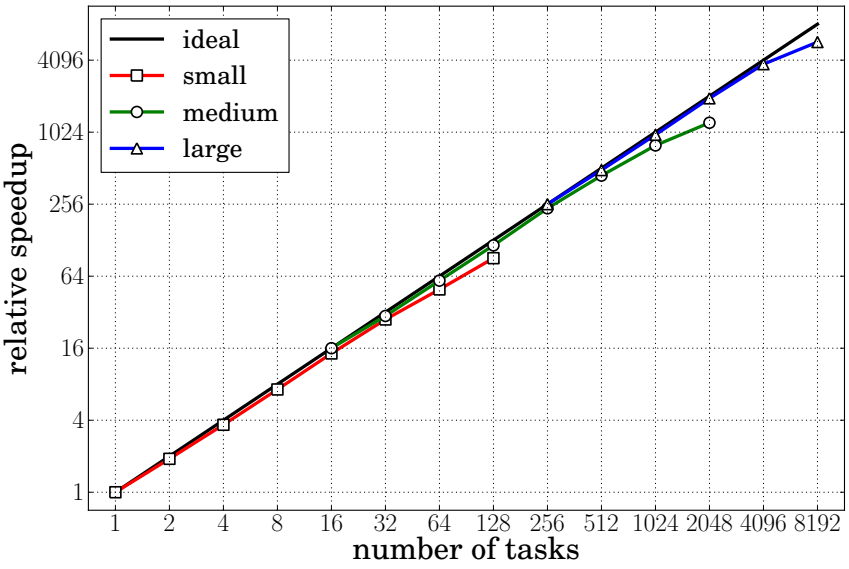


Figure 4.13: Hybrid scaling on an IBM Blue Gene/P system for various number of particles on the spherical vortex sheet, strong scaling for each dataset. Small: $N = 1{\cdot}10^5$, medium: $N = 1.6 \cdot 10^6$, large: $N = 2.56 \cdot 10^7$.

per tasks, but also to the number of tasks itself, i.e. the speedup flattens out earlier for higher numbers of tasks.

To analyze the scaling in more detail, Figure 4.14 shows the timings for the larger dataset for each of the three steps described in Section 4.1. It becomes clear that these steps behave very differently for a large number of tasks. While the time consuming and dominant tree traversal shows perfect strong scaling, domain decomposition and tree construction are clearly responsible for the performance saturation. Since these steps do not scale at all for large numbers of tasks or low numbers of particles per tasks, respectively, they start to dominate the total iteration time for 8192 tasks. Both steps include collective operations: the domain decomposition during parallel sorting, the tree construction during the branch exchange process. Furthermore, the amount of data exchanged during these operations increases with the number of tasks. In particular, the branch structure becomes an issue at high task numbers. The branch level, i.e. their depth in the tree relative to root and thereby the number of branches increases during a strong scaling scenario, i.e. for a fixed problem size with variable number of tasks. In addition, the structure of fill nodes expands as well, so that this step takes more time. The consequences and costs arising from the branch structure are therefore responsible for the scaling failure of the tree construction step. During the analysis of the theoretical runtime complexity we have already seen that the overall complexity contains a problematic

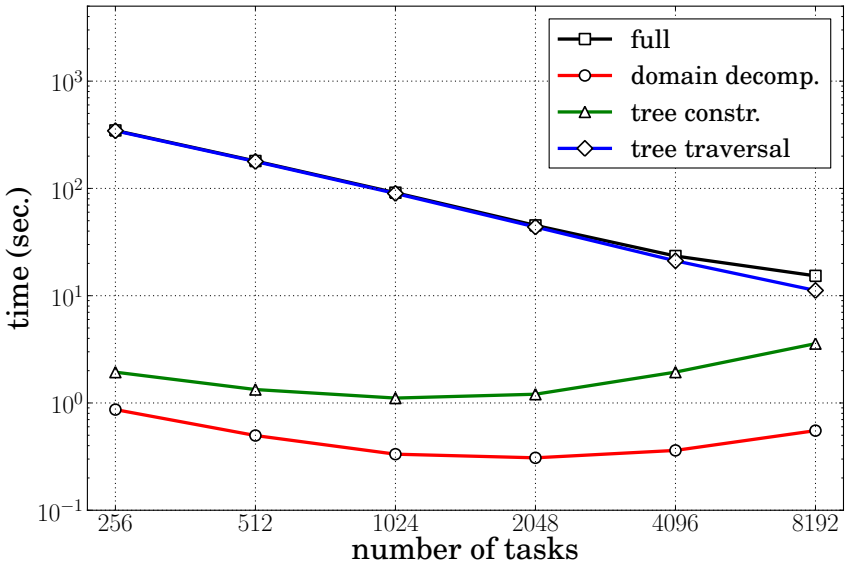


Figure 4.14: Detailed timings for the spherical vortex sheet on an IBM Blue Gene/P system for $2.56 \cdot 10^7$ particles, split into the three main steps: domain decomposition, tree construction and tree traversal.

$P \log P$-term, and the practical scaling tests verify this issue, limiting efficient use of the code to less than 8192 MPI tasks at the moment.

While parallel sorting and reordering of the particles during domain decomposition is a necessary and inevitable part of tree codes and is already implemented very efficiently (see [113] for details), it is more important to see how branch and global node structure affect the actual force summation. A detailed investigation with the purely Coulombian tree code PEPC-E and a homogeneous setup in a unit cube revealed the following interesting aspect. While the number of branches increased significantly for increasing number of tasks, they had virtually no impact on the actual force summation. For 16384 MPI tasks on JUGENE only 0.73% of the whole global node structure including the non-local branches was used for at least one interaction. This suggests that reduction of these structures may provide a promising route to optimizing efficiency, memory consumption and scaling.

Despite the current limitations, making effective use of 8192 MPI tasks and being able to simulate more than 2×10^8 particles the underlying Barnes-Hut tree code is highly competitive among currently available tree codes. In the next section we review other algorithmic approaches with their applications and place our tree code in the context of their capabilities.

4.2.3 A comparative overview of other vortex codes

For numerical vortex methods many different algorithms are currently being developed. Naturally, many codes are purely particle-based, but recently, the mesh-based vortex-in-cell method has become an interesting alternative. In this section, we review some competing codes and highlight their differences and capabilities.

Firstly, we focus on two Barnes-Hut tree codes, which are directly applied to vortex particle methods as well. In Section 3.2.5, we have already introduced one of them [55, 99]. Here, the Gaussian smoothing kernel is used, but the multipole expansion (or an approximation by a simple ρ^{-1}-potential) is applied only for distances larger than 5σ. This choice of cutoff in this code is inherently linked to the data-dependent multipole acceptance criterion by the user-defined relative error. Within a radius of 5σ direct interactions are used. Simulating the spherical vortex sheet and vortex ring dynamics they require remeshing as well, exploiting the oct-tree data structure instead of using the sorting algorithm. In [55], they show simulations with 10^6 particles using up to 512 processor with very good scaling results. However, the farfield-nearfield definition makes this approach prone to clustered distributions and therefore $\mathcal{O}(N^2)$-complexity. Furthermore, the multipole expansion for exponential and error function are expensive in terms of computational costs. The native expansion of algebraic kernels as implemented in our work solves both issues: while easier and faster to derive during a simulation there is no need to define a cutoff in addition to the MAC.

Another tree-based algorithm is presented in [101]. In contrast to the Gaussian kernel, the authors make use of the zeroth order algebraic kernel and apply this method to vortex sheet dynamics. As noted in [17], the usage of this kernel may not lead to a convergent method in three spatial dimensions and should be applied with great care only. They use a data-dependent MAC as well, but do not implement a scheme to ensure the overlap condition. Their simulations run with up to 10^5 particles, but not in parallel. Again, by using the generalized class A_σ as basis for the multipole expansion theory, our tree code does not share the problems induced by the low order kernel, since the expansion and convergence result can be applied to algebraic kernels of arbitrary order. Furthermore, our code is able to simulate two orders of magnitude more particles by exploiting distributed memory systems effectively.

A similar concept to Barnes-Hut tree codes is the application of Fast Multipole Methods. One parallel implementation in the context of vortex particle methods is reported in [117]. For 2D Lamb-Oseen vortex simulations, the authors use the Gaussian kernel with ρ^{-1}-approximation in the FMM farfield and direct interactions in the nearfield. Since the FMM has an intrinsic distinction between far- and nearfield, their approximation by ρ^{-1} seems to be more well-grounded. However, the inconsistency of the approximation in the farfield remains. The code presented in this publication scales very well up to 64 processors and is able to handle 10^7 particles. The authors already mention the fact that further scalings would require additional improvements. Although not explicitly stated, we assume that instead of remeshing, radial

basis function interpolation is applied, requiring $\mathcal{O}(N)$ operations per evaluation point [54]. Their results and applications are limited to two-dimensional examples.

The codes reviewed so far are all versions of pure mesh-free algorithms and their capabilities in number of particles and usage of parallel architectures are more or less limited. For algorithms dealing with more than 10^9 particles and thousands of processors, a fundamentally different approach is commonly chosen: the vortex-in-cell (VIC) algorithm [64]. Based on the assumption that remeshing is inherently required and performed regularly in vortex methods, VIC codes consequently demand the presence of a mesh structure and rely on fast Poisson solvers [118] on regular meshes for computing stream function and velocity. A comprehensive description of a VIC workflow can be found in [70], coupled with an FMM code for boundary conditions. The current state-of-the-art VIC algorithm is presented in [119]. In this publication, the code is used for simulation of aircraft wakes with periodic boundary conditions. The simulation box is discretized with $2048 \times 1024 \times 786$ mesh nodes and 1.6×10^9 particles, showing good scalability and excellent runtimes up to 16384 processors on an IBM Blue Gene/L system.

Despite the impressive achievements reported in this and other VIC publications, the main drawback of these mesh-based algorithms remains: the excessive use of a mesh structure requires a very advanced spatial adaptation concept or memory-consuming and inefficient mesh sizes. While in tree codes the costs for tree construction and management as discussed in Section 4.2.2 still leave room for improvement, the mesh-free character induces an intrinsic adaptivity without the need for further efforts. Furthermore, particle-based algorithms can handle unbound problems much more easily, while the need for an underlying mesh requires further implications for these setups.

Next, we show another application of our tree code, which is often used in three-dimensional vortex codes: the fusion of two vortex rings. This configuration inherently requires viscous processes and together with remeshing this example demonstrates the capabilities of our parallel tree code for vortex particle methods.

4.2.4 An application: vortex ring dynamics

Vortex ring dynamics are an interesting choice to use three-dimensional vortex methods in physical investigations. We have already seen one example of a vortex ring in Section 4.2.2. Experimental setups of two side-by-side identical viscous vortex rings with circular vorticity fields revealed a fundamental physical process: the fusion reconnection. Since experiments, e.g. in water [120] or air [121], are hard to set up and investigate, only numerical simulations are able to shed light on topological details of these types of phenomena. A very comprehensive study of vortex ring dynamics using a mesh-based spectral method can be found in [122] and we refer to this work for physical implications and interpretations of the upcoming simulation.

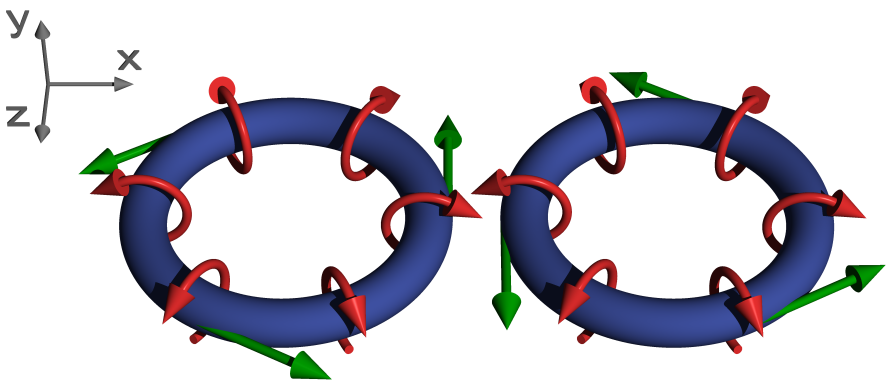


Figure 4.15: The cores of two side-by-side vortex rings. Schematically, straight arrows depict the tangential vorticity vectors, circular arrows indicate the movement of the rings' particles according to the velocity vectors.

Since the fusion process of vortex rings is a viscous process, numerical simulations require the treatment of the diffusion term of the vorticity-velocity equation. Remeshing is mandatory as well for these complex geometries and long timescales, so that we can conveniently use the modified remeshing approach of [76] as presented in Section 2.2.4. For reference, we choose the initial conditions in the style of [76, 100]. One major challenge in the numerical simulation of vortex rings is the initial particle discretization of the toroidal geometry. We review and use the idea given in [32]. First, we discretize a single circular segment in the (x, y)-plane, starting from the center. These particles are attached with a vorticity vector in z-direction. Next, these two-dimensional segments are rotated, translated and copied to form a ring of segments with vorticity vectors along the ring tubes. Therefore, four parameters determine the construction of a single vortex ring:

- the radius r_c of each segment,
- the number n_c of circles in each segment ($n_c = 0$ for a single particle at the center),
- the number N_φ of segments,
- and the radius R of the complete ring, measured from the center of the ring to the center of each segment.

Furthermore, we have to choose the vorticity distribution inside each segment. Classically, the function

$$\omega(x) = \omega_0 \exp\left(-\left(\frac{|x|}{r_c}\right)^2\right)$$

is used, where x is the location inside the segment. Finally, the second vortex ring is a straight copy of the first one, with their centers separated by s. Following the setups of [32, 76], we use $N_\varphi = 126$ segments of radii $r_c = 0.4$ with $n_c = 7$ circular layers and maximum vorticity

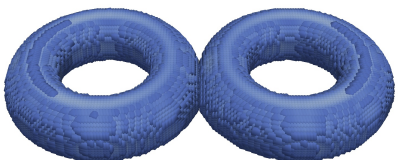
$\omega_0 = 23.8$ and form two rings with radii $R = 1$ and separation $s = 3.65$. Each segment has $1+4n_c(n_c+1) = 225$ particles, so that both rings together initially have $N = 56700$ particles. By transformation, the rings are located in the (x, z)-plane with $y = 0$. In Figure 4.15 both rings are schematically depicted. Here, straight arrows show the tangential vorticity vectors, while circular arrows indicate the movement of the rings' particles according to the velocity vectors.

For the viscous simulation with our tree code, the second order algebraic kernel with a fixed core size $\sigma = 0.2$ is used and evaluated with the Bmax-MAC and $\theta = 0.45$. We apply the modified remeshing operator, exploiting viscous splitting every two time steps with interpolation size $h = 0.05$ and a population control threshold of $|\omega(x)| \leq 0.01$. The third order Runge-Kutta time integrator uses $\Delta t = 0.05$ and the viscosity is set to $\nu = 0.01$.

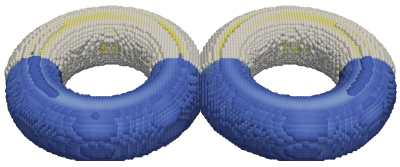
In Figure 4.16, we show the development of the two vortex rings. In the left column, only particles with $|\omega(x)| \in [3, 23.8]$ are depicted. In the right column, we show half of the particles with $|\omega(x)| \in [0, 3)$ to show the impact of remeshing on the simulation and the number of particles. We start the simulation with $N = 56700$ particles discretizing the two vortex rings at $t = 0.0$. Despite a rigorous population control, due to remeshing and diffusion the number of particles grows to $N = 883215$ at $t = 9.8$ after 196 time steps. Starting from the (x, z)-plane with $y = 0$, the rings travel in the positive y-direction and move towards the $x = 0$ plane, being attracted by mutual induction. This can be seen in Figures 4.16a to 4.16b and Figures 4.16e to 4.16f, respectively. Induced by the collision of the inner cores, vortex bridges are formed, as depicted in Figures 4.16c to 4.16d and Figures 4.16g to 4.16h, respectively. During fusion the circulation of the inner cores decreases while the circulation inside the bridges increases rapidly. Furthermore, during this process large gradients of vorticity appear, but they get annihilated by vortex diffusion. Eventually, the two rings are merged to a single yet disturbed vortex ring, as Figures 4.16d and 4.16h show.

In Figure 4.17, we compare the dynamics of viscous and inviscid vortex rings. The right column again show half of the particles with $|\omega(x)| \in [0, 3)$ for the viscous simulation with $\nu = 0.01$. To compare the behavior of the vorticity with an inviscid configuration, the left column depict the dynamics of the same setup, but with $\nu = 0.0$, again visualizing particles with $|\omega(x)| \in [0, 3)$. Dark colors indicate high vorticity magnitudes. We can clearly see how the diffusion process damps the vorticity field and extends its support. Moreover, in the inviscid configuration the bridges are merged (see Figure 4.17g), but do not split apart, as Figure 4.17h indicates. Therefore, an inviscid simulation does not lead to a single merged vortex ring. For more on the physical details of these phenomena we refer to [32, 122].

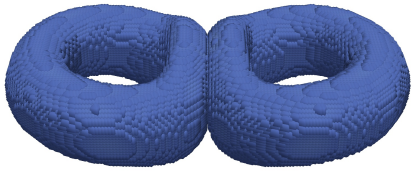
With the ability of simulating vortex rings and their interactions, the code is capable of covering a broad field of numerical experiments, e.g. ring propagation and stability [18] or knot configurations [32].



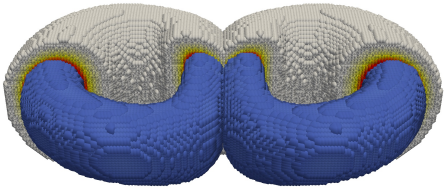
(a) Time $t = 0.16$, $3 \leq |\omega| \leq 23.8$



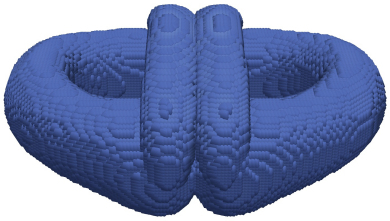
(e) Time $t = 0.16$, added remeshing particles



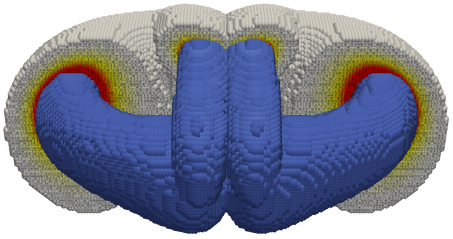
(b) Time $t = 3.2$, $3 \leq |\omega| \leq 23.8$



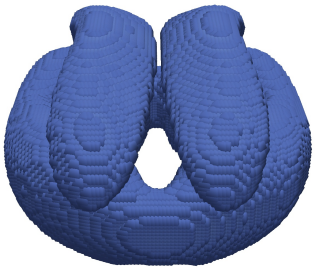
(f) Time $t = 3.2$, added remeshing particles



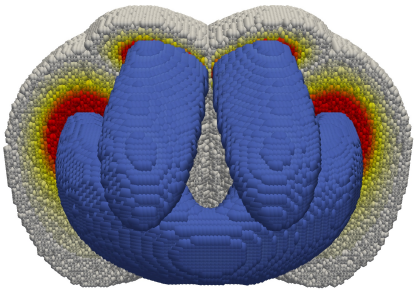
(c) Time $t = 6.4$, $3 \leq |\omega| \leq 23.8$



(g) Time $t = 6.4$, added remeshing particles

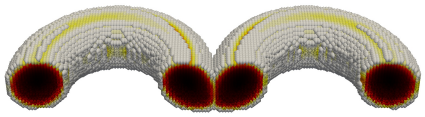


(d) Time $t = 9.8$, $3 \leq |\omega| \leq 23.8$

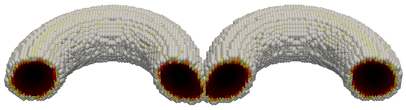


(h) Time $t = 9.8$, added remeshing particles

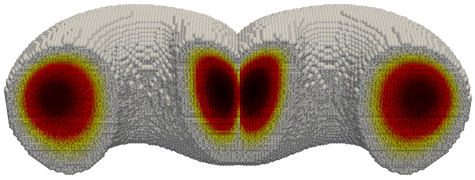
Figure 4.16: Fusion of two vortex rings. Left column: particles with $|\omega(x)| \in [3, 23.8]$, right: same threshold, but additionally one half of the other particles. Discretization parameters: $N_\varphi = 126$, $n_c = 7$, $r_c = 0.4$, $R = 1$, $s = 3.65$. Simulation parameters: $\omega_0 = 23.8$, $\sigma = 0.2$, $h = 0.05$, $\nu = 0.01$, $\Delta t = 0.05$. Threshold is $|\omega(x)| \leq 0.01$, Bmax-MAC uses $\theta = 0.45$.



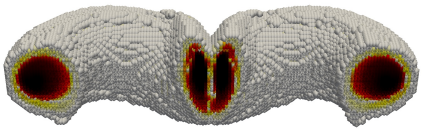
(a) Time $t = 0.16$, viscous setup



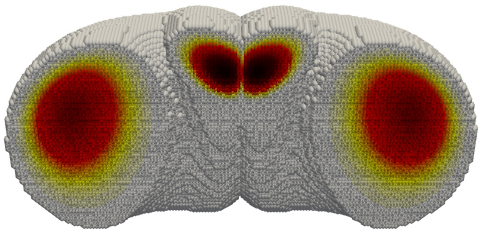
(e) Time $t = 0.16$, inviscid setup



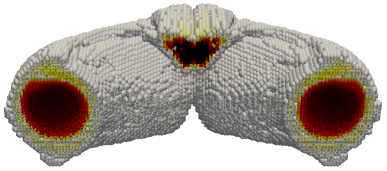
(b) Time $t = 3.2$, viscous setup



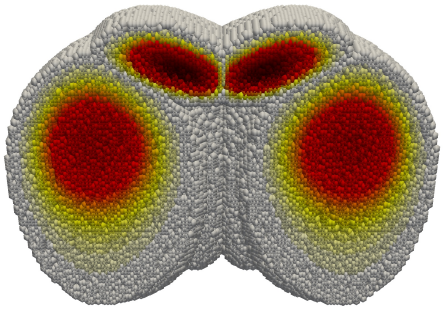
(f) Time $t = 3.2$, inviscid setup



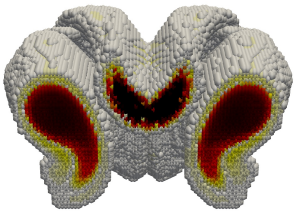
(c) Time $t = 6.4$, viscous setup



(g) Time $t = 6.4$, inviscid setup



(d) Time $t = 9.8$, viscous setup



(h) Time $t = 9.8$, inviscid setup

Figure 4.17: Fusion of two vortex rings, one half of all particles. Left column: viscous setup with $\nu = 0.01$, right: inviscid setup with $\nu = 0.0$. Discretization parameters: $N_\varphi = 126$, $n_c = 7$, $r_c = 0.4$, $R = 1$, $s = 3.65$. Simulation parameters: $\omega_0 = 23.8$, $\sigma = 0.2$, $h = 0.05$, , $\Delta t = 0.05$. Threshold is $|\omega(x)| \leq 0.01$, Bmax-MAC uses $\theta = 0.45$. Dark particles indicate high vorticity magnitude.

4.3 Summary

In this chapter, we have described and analyzed a new implementation of a parallel Barnes-Hut tree code for regularized vortex particle methods, based on the mathematical theory of algebraic kernels and multipole expansions. Figure 4.18 updates the introductory choice between FMM and Barnes-Hut tree code in favor of the latter approach. A comprehensive description and a detailed scaling analysis of the parallel Barnes-Hut tree code PEPC and its adaptation to regularized vortex particle methods have been the topic of this chapter.

The description of the parallelization scheme already indicated that global communication induced by the branch concept is a crucial part of our tree code. This observation has been verified and analyzed by benchmarking the code using a spherical vortex sheet. The branch structure accounts for the intrinsic requirement of global information exchange in long-range dominated problems. Here, nearest-neighbor communication patterns are not sufficient and although the current concept and its implementation leaves room for promising improvements, it provides a functional and reliable way of handling non-local information exchange in parallel tree codes. We have seen that both theoretical complexity and practical scaling analysis showed a P-dependency, which also has a severe impact on memory requirements. Despite these obstacles, the parallel Barnes-Hut tree code presented in this work scales nicely up to 4096 MPI ranks and is able to perform vortex particle simulations with millions of elements routinely.

While particle-based codes in the literature appear to have severe problems with many-million particle simulations on parallel architectures, the vortex code presented in this work is able to solve problems with more than 10^8 particles on thousands of processors. Concerning problem size and parallelism the code therefore covers applications, where mesh-based methods like vortex-in-cell algorithms are predominant. Using a novel and flexible remeshing approach, which exploits parallel sorting within the domain decomposition scheme, we are able to conserve the mesh-free character of particle-based methods without affecting the overall complexity of our tree code. With frequent remeshing, even complex problems and setups e.g. in the broad field of vortex ring dynamics can be performed with excellent results.

In the final chapter, we will summarize the results of this work and give an outlook to future directions in the field of multipole-based algorithms for regularized vortex particle methods.

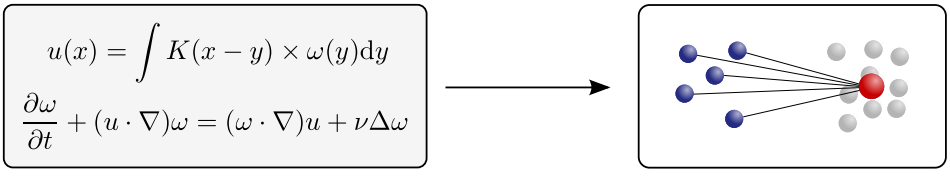


Figure 4.18: The transition from analytic equations to Barnes-Hut tree codes.

5 From theory to practice: a conclusion

5.1 Summary

We have started this work with a reformulation of the classical Navier-Stokes equations. The resulting vorticity-velocity equation and an explicit expression for the velocity field are the basis for vortex methods. However, using this approach for a numerical simulation of vortex-driven flows, the analytical form of the velocity field has to be discretized. In Chapters 2 to 4 we have presented one possible way to obtain a rapid numerical scheme, which is capable of simulating complex systems such as vortex rings. The choices we have made on this way defined the content of each of these chapters: mesh- or particle-based discretization, cutoff- or multipole-based summation, and FMM or Barnes-Hut tree code. We recapitulate the results of this work in the following, supported by the scheme in Figure 5.1.

Discretizing the vorticity field using particles has been the first decision and the consequences were the topic of Chapter 2. The great advantage of particle-based schemes are the intrinsic, efficient adaptivity and the straightforward treatment of open boundaries at infinity, while mesh-based methods often require very complex strategies for achieving both attributes. We

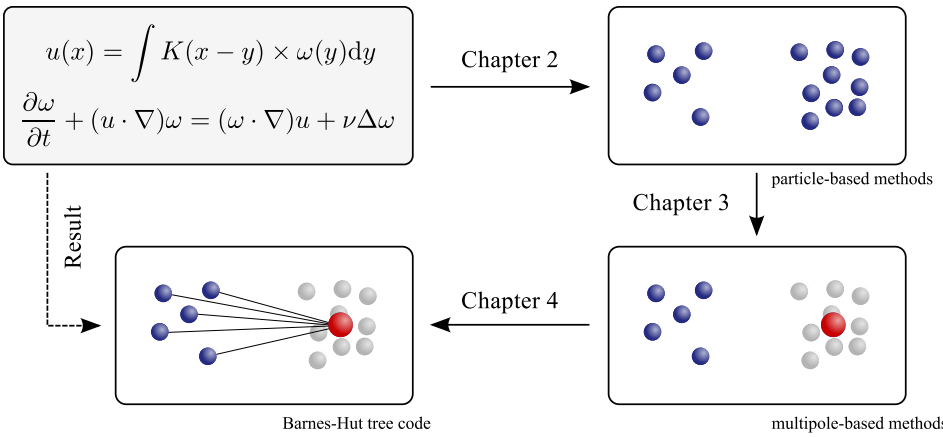


Figure 5.1: From vorticity-velocity equation and Biot-Savart formulation to a tree code.

have seen that using not only point particles but rather regularized, smeared vortex blobs leads to a consistent convergence result (see Theorem 2.1.8). The order of the smoothing kernels and their core size determine the speed of convergence, which abolishes the limitation of the point particle scheme to second order convergence. However, it is well-known that the use of regularized smoothing kernels implies a strong requirement on the underlying particle set. The overlap criterion, especially required for long-term simulations, has to be taken into account carefully for any practical simulation. One convenient way of restoring a distorted particle set is given by the method of remeshing. Frequent interpolation onto a regular particle field with predefined interparticle distances allows for accurate and well-grounded long-term simulations. Moreover, we have seen that simple modifications of the interpolation kernels can extend inviscid codes to treat diffusion as well. This interesting aspect makes the remeshing approach even more appealing, since other viscous methods often require additional efforts.

However, the remeshing concept firstly induces an underlying mesh structure, thus canceling the advantages of particle methods. This drawback has to be considered and addressed by a numerical implementation. Smart data structures and algorithms can prevent this, as we have seen in Section 4.1.4. In our novel parallel remeshing approach, each source particle creates its own local interpolation targets and by using efficient parallel sorting the duplicates are combined and eliminated subsequently. It is a native extension to sorting-based domain decomposition techniques and does not rely on expensive mesh structures. The resulting target particles cover only areas of interest, thus being the minimal number of particles required for the next time step. Preserving the mesh-free character of particle methods, this new approach provides an efficient and flexible tool to ensure the overlap condition. Furthermore, its theoretical complexity is limited by the complexity of parallel sorting.

Theoretical complexity is the reason for the second decision. The use of particles implies an N-body problem for computing the velocity and the vorticity update, since the effect of smoothing kernels is not limited to short-distance particles. A naive implementation with N particles leads to a $\mathcal{O}(N^2)$-scheme, preventing the usability of such an implementation with high numbers of N. In contrast to cutoff-based concepts, multipole-based approaches yield a rigorous mathematical background and effective error control to reduce the complexity to at least $\mathcal{O}(N \log N)$. To apply the concept of multipole expansions to regularized vortex particle methods, the smoothing kernels have to be analyzed carefully. We introduced a novel class A_σ of algebraic smoothing kernels in Definition 3.1.1. This class and the generalization of algebraic kernels itself allows for the construction of higher order kernels, which satisfy the requirements of the Convergence Theorem 2.1.8 (see Theorems 3.1.3 and 3.1.9).

Moreover, we have shown that each member of this class is a linear combination of much simpler functions. The Decomposition Theorem 3.1.5 considerably simplifies the analysis of generalized algebraic kernels by reducing the class A_σ to a component class D_σ. In particular, this reduction formed the basis for the analysis of multipole expansions in the context of algebraic kernels. We have seen in Theorem 3.2.8 that for each function of D_σ the remainder

of the multipole expansion can be bounded using the distance between particle and cluster and the cluster size. As a nice consequence, this theorem also extends well-known results for ρ^{-n}-kernels to their regularized counterparts. Using the same linear combinations of the Decomposition Theorem, the convergence results yield a comprehensive theory for generalized algebraic kernels. The concept of multipole expansions can now be applied to the regularized vortex particle method on a well-grounded mathematical basis.

The central advantage for a numerical implementation of multipole expansions is the separation of multipole moments and the actual differentiation of the kernel. The moments only depend on the cluster and its members themselves, so that they can be precomputed. During the expansion only the current particle and the center-of-charge of the cluster are relevant. This clear separation yields the basis for the Barnes-Hut tree code. Being the result of the third decision, the parallel Barnes-Hut tree code PEPC has been rewritten and adapted to account for algebraic regularized particle systems in Chapter 4. An inevitable part of such tree codes is a domain decomposition scheme. This is done efficiently by parallel sorting algorithms, so that our novel remeshing implementation can be seen as a native extension of domain decomposition, integrating nicely into existing tree codes. We have described the parallelization strategy of the tree code and its theoretical complexity in Section 4.1. Using a spherical vortex sheet as setup, we have shown scalability on an IBM Blue Gene/P system up to 8192 cores, thus allowing rapid simulations with up to 10^8 particles. The usability of the code has been demonstrated with a complex setup of two merging vortex rings in Section 4.2.4.

We have started this work with a theoretical approach for treating the Navier-Stokes equations. Using regularized particles in Chapter 2, we have reviewed the main concepts of vortex particle methods, which are required for a first and naive numerical implementation. The introduction of a new class of algebraic kernels in Chapter 3 has paved the way for fast multipole-based methods in the context of regularized vortex particle methods. Finally, we have adapted a fully parallelized Barnes-Hut tree code in Chapter 4, which puts the theoretical concepts of the preceding chapters into practice.

Finally, we highlight topics of further interest in the following, concluding section.

5.2 Outlook

The theory presented in this work satisfies its purpose of yielding a reliable basis for a fast, accurate summation technique for regularized vortex particle methods in unbounded domains. Using the parallel Barnes-Hut tree code PEPC as fundament, the algorithm developed here can routinely perform vortex particle simulations with millions of elements. We conclude this work with a short overview of three topics, which may enrich and extend theory, algorithm and fields of application even further.

Bounding the box – At the very beginning of this work we have restricted ourselves to unbounded domains only. Of course, this is a severe limitation with respect to practical applications. The treatment of both inviscid and viscous boundary conditions in vortex methods is a topic of active and versatile research (see [13] and references therein for a detailed mathematical and application-oriented overview). The presence of solid boundaries has significant impact on many facets of vortex methods: remeshing may require non-regular grids [123], particle strength exchange can use image particles [69] or one-sided kernels [124]. On the other hand, periodic boundary conditions play a vital role in numerical vortex simulations as well. Using a mesh structure, vortex-in-cell (VIC) codes include this type of boundary condition directly [119], while purely particle-based algorithms often make use of the so-called Ewald summation technique [125, 126]. This technique yields an $\mathcal{O}(N \log N)$-scheme for computing the potentially infinite number of contributions in periodic particle simulations. However, in the context of Fast Multipole Methods (FMM), this complexity is still unfavorable. In [127, 128], this scheme is replaced with a renormalization approach, yielding $\mathcal{O}(N)$ complexity. Based on the spherical multipole expansion of the ρ^{-1}-kernel, this renormalization fits nicely into the context of FMM implementations [105]. In principle, this approach can be extended to ρ^{-n}-kernels and their regularized counterparts, i.e. to the class D_σ.

Looking left and right – The method of renormalization for periodic boundaries is applicable for all kernels that can be used with the FMM. This is due to the fact that the operators required for renormalization are basically the same as required for the down-shift and translation within the FMM. To the best of our knowledge, these operators have not been investigated for the full class D_σ so far. A successful adaptation would embed the theory of the classes D_σ and A_σ into the context of FMM and renormalization simultaneously. Furthermore, it might be interesting to extend the present theory to benefit from an efficient recurrence relation for the coefficients of the Taylor expansion, as shown in [101]. The recursion has been already adapted to the second order algebraic kernel [100] and we can assume that a further extension to the class D_σ is straightforward.

Optimizing the interplay of theory and practice – Currently, the tree code is optimized for using the second order algebraic kernel only. With the theory of D_σ at hand, the code can be extended to compute multipole expansions for each member of D_σ. Using the Decomposition Theorem and multiple calls of the expansion routines, this has the great advantage that the code then covers the full classes A_σ and $\bar{A}_\sigma$, including ρ^{-n}-kernels and algebraic kernels of arbitrary order. Especially the last consequence can be of great interest. As noted in [49, 58], the rezoning scheme of Section 2.2.1 requires smoothing kernels ζ_σ of higher order to minimize unwanted diffusive errors. Furthermore, we have seen that this procedure requires $\mathcal{O}(NM)$ operations for N source and M target particles. Since we have $\zeta_\sigma \in A_\sigma$ for algebraic kernels, the present theory of this work gives us a very powerful tool to apply the concept of multipole expansions to rezoning. Therefore, a tree code covering the class D_σ can be easily extended to use efficient rezoning, thus reducing the complexity to $\mathcal{O}(M \log N)$. Additionally, Theorem 3.2.11 has shown that kernels η_σ within the particle strength exchange approach (see Section 2.2.3) can be chosen as members of A_σ, so that this summation can be

accelerated using multipole expansions as well. However, the multipole moments do not coincide with the classical ones. More effort is required to integrate this scheme in the context of tree codes.

The comprehensive, yet compact formulation of algebraic kernels using the class A_σ and its component class D_σ has been developed as a powerful and elegant foundation of the theory of multipole expansions in this work. We now see that it is also an appealing source for many promising and challenging extensions.

Acknowledgments

This thesis would not have been possible without the help, faith and support of many people. I would like to thank Prof. Dr. Rolf Krause for his advice and patient supervision. In fruitful discussions and pleasant atmosphere he guided me and was of great help, not only but also concerning scientific questions. Moreover, it is an honor for me to thank Prof. Dr. Andreas Frommer and the research group Applied Computer Sciences at Wuppertal University for their "friendly takeover" and extensive support.

I am very grateful that this work could take place at Jülich Supercomputing Centre. I would like to express my sincere gratitude to Prof. Dr. Dr. Thomas Lippert for giving me the opportunity and support to work on this thesis at his institute. His open doors and mind eased some troubling situations. This is also true for Dr. Paul Gibbon, my adviser at JSC. He gave me encouraging and fascinating insights into the broad field of his work and I would like to thank him for his dedication, especially concerning his "mathematical tolerance" as a physicist. There are too many more helpful persons at JSC to mention all. However, I would like to explicitly acknowledge the great support of the technical and administrative JSC staff, providing tireless work on and against many facets of bureaucracy.

My sincere gratitude goes to Ivo Kabadshow, Natalie Schröder and Mathias Winkel for their rigorous proof-reading with sharp pencils and clear words. Being my Ph.D. colleagues at JSC, they enriched my time there with stimulating discussions and invaluable assistance and they became friends beyond our scientific work.

In random order and for myriad reasons I would like to thank Matthias, Kerstin, Sascha, Thomas and Marion. Last but by no means least, I owe my deepest gratitude to Jessica and my parents for their love, their faith and their absolute support, not only during my wanderings.

> *"I am fain to compare myself with a wanderer on the mountains who, not knowing the path, climbs slowly and painfully upwards and often has to retrace his steps because he can go no further – then, whether by taking thought or from luck, discovers a new track that leads him on a little till at length when he reaches the summit he finds to his shame that there is a royal road by which he might have ascended, had he only the wits to find the right approach to it."*
>
> Hermann Ludwig Ferdinand von Helmholtz, 1906 [129]

A Vector calculus notation and identities

For clarity we state the basic operators in more detail and cite some important identities.

Definition A.1 (Vector-valued differential operators)
For vectors $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $n, m \in \mathbb{N}$ we define

a) the *gradient* of u as ∇u with

$$\nabla u := \begin{pmatrix} \frac{\partial u_1}{\partial x_1} & \cdots & \frac{\partial u_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial u_m}{\partial x_1} & \cdots & \frac{\partial u_m}{\partial x_n} \end{pmatrix} \in \mathbb{R}^{m \times n},$$

b) the *divergence* of u as $\nabla \cdot u$ for $m = n$ with

$$\nabla \cdot u := \sum_{i=1}^{n} \frac{\partial u_i}{\partial x_i} \in \mathbb{R}^1,$$

c) the *curl* of u as $\nabla \times u$ for $m = n = 2$ with

$$\nabla \times u := \frac{\partial u_2}{\partial x_1} - \frac{\partial u_1}{\partial x_2} \in \mathbb{R}^1$$

and for $m = n = 3$ with

$$\nabla \times u := \left(\frac{\partial u_3}{\partial x_2} - \frac{\partial u_2}{\partial x_3}, \frac{\partial u_1}{\partial x_3} - \frac{\partial u_3}{\partial x_1}, \frac{\partial u_2}{\partial x_1} - \frac{\partial u_1}{\partial x_2} \right)^T \in \mathbb{R}^3,$$

and finally

d) the *vector Laplacian* of u as Δu with

$$\Delta u := (\Delta u_1, \ldots, \Delta u_m)^T = \left(\sum_{i=1}^{n} \frac{\partial^2 u_1}{\partial x_i^2}, \ldots, \sum_{i=1}^{n} \frac{\partial^2 u_m}{\partial x_i^2} \right)^T.$$

Using these elementary definitions, we obtain the following identities.

Theorem A.2 (Vector identities)
For vectors $u, v \in \mathbb{R}^3$ we have

$$\nabla \cdot (\nabla \times u) = 0, \tag{A.1}$$
$$\nabla \times (\nabla (u \cdot v)) = 0, \tag{A.2}$$
$$u \times u = 0, \tag{A.3}$$
$$u \times v = -v \times u. \tag{A.4}$$

Furthermore, we have

$$\nabla (u \cdot v) = (u \cdot \nabla) v + (v \cdot \nabla) u + u \times (\nabla \times v) + v \times (\nabla \times u), \tag{A.5}$$
$$\nabla \times (v \times u) = v(\nabla \cdot u) - u(\nabla \cdot v) + (u \cdot \nabla) v - (v \cdot \nabla) u, \tag{A.6}$$
$$(\nabla \times u) \times v = (\nabla u - (\nabla u)^T) \cdot v, \tag{A.7}$$
$$\Delta u = \nabla (\nabla \cdot u) - \nabla \times (\nabla \times u), \tag{A.8}$$
$$\nabla \times (\nabla \times u) = \nabla (\nabla \cdot u) - \Delta u, \tag{A.9}$$

where

$$(u \cdot \nabla) v = \left(\sum_{i=1}^{3} u_i \frac{\partial v_1}{\partial x_i}, \sum_{i=1}^{3} u_i \frac{\partial v_2}{\partial x_i}, \sum_{i=1}^{3} u_i \frac{\partial v_3}{\partial x_i} \right)^T,$$
$$v(\nabla \cdot u) = \left(v_1 \sum_{i=1}^{3} \frac{\partial u_i}{\partial x_i}, v_2 \sum_{i=1}^{3} \frac{\partial u_i}{\partial x_i}, v_3 \sum_{i=1}^{3} \frac{\partial u_i}{\partial x_i} \right)^T.$$

Proof. Proofs of these identities can be found in [3, 20, 130]. □

Finally, the multi-index notation is frequently used throughout this work. The next definition highlights its impact on vector calculus.

Definition A.3 (Multi-index notation)
A multi-index β is an n-tupel $(\beta_1, \ldots, \beta_n)$ with $\beta_i \in \mathbb{N}_0$, $i = 1, \ldots, n$. We define

a) the factorial of β as $\beta! = \beta_1! \cdot \ldots \cdot \beta_n!$ and

b) the norm of β as $|\beta| = \beta_1 + \ldots + \beta_n$.

Furthermore, for a vector $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ we define x^β as $x^\beta = x_1^{\beta_1} \cdot \ldots \cdot x_n^{\beta_n}$.

B Sobolev spaces

In this appendix, we review some functional analytic ideas and concepts. More on this topic can be found in [50, 51] or in any other standard book on functional analysis. We start with the definition of spaces of continuously differentiable functions.

Definition B.1 (Differentiable functions)

a) For a multi-index $\beta = (\beta_1, \ldots, \beta_d)$ (see Definition A.3) and $x \in \mathbb{R}^d$ we define the *β-th order partial derivative* of a function f as

$$D^\beta f(x) = \frac{\partial^{|\beta|} f(x)}{\partial x_1^{\beta_1} \ldots \partial x_d^{\beta_d}}.$$

b) For $\Omega \subset \mathbb{R}^d$ the space $C(\Omega)$ consists of all real-valued functions f that are continuous on Ω. Furthermore, for $m \in \mathbb{N}_0$ we define

$$C^m(\Omega) := \{f \in C(\Omega) : D^\beta f \in C(\Omega) \text{ for } |\beta| \leq m\},$$
$$C^\infty(\Omega) := \bigcap_{m=0}^{\infty} C^m(\Omega) = \{f \in C(\Omega) : f \in C^m(\Omega) \text{ for all } m \in \mathbb{N}_0\}.$$

c) For $\operatorname{supp}(f) := \overline{\{x \in \Omega : f(x) \neq 0\}}$ we set

$$C_0^\infty(\Omega) := \{f \in C^\infty(\Omega) : \operatorname{supp}(f) \subsetneq \Omega\}.$$

The next step is the definition of Lebesgue spaces $L^p(\Omega)$ and their corresponding norms. Furthermore, local integrability is covered.

Definition B.2 (Lebesgue spaces)

a) We define the *Lebesgue space* $L^p(\Omega)$, $\Omega \subset \mathbb{R}^d$, $p \in [0, \infty)$ as the space of all measurable functions $f : \Omega \to \mathbb{R}$ with

$$||f||_{L^p(\Omega)} := \left(\int_\Omega |f(x)|^p \right)^{\frac{1}{p}} < \infty.$$

b) For $p = \infty$ the *Lebesgue space* $L^\infty(\Omega)$ is identified as the space of all measurable functions $f : \Omega \to \mathbb{R}$ with

$$||f||_{L^\infty(\Omega)} := \sup_{x \in \Omega \setminus \Omega'} |f(x)| < \infty \text{ for } \mathrm{meas}(\Omega') = 0.$$

c) For $1 \leq p < \infty$ we call a function f *locally p-integrable*, if for every $x \in \Omega$ there is an open neighborhood Ω' of x with $\overline{\Omega'} \subset \Omega$ and $f \in L^p(\Omega')$. The space of all such functions is denoted as $L^p_{\mathrm{loc}}(\Omega)$.

With these spaces at hand we now extend the classical definition of derivatives.

Definition B.3 (Weak derivatives)
For a non-empty set $\Omega \subset \mathbb{R}^d$ and $f, g \in L^1_{\mathrm{loc}}(\Omega)$ we call g the *weak β-th derivative* of f if

$$\int_\Omega f(x) D^\beta \phi(x) \mathrm{d}x = (-1)^{|\beta|} \int_\Omega g(x)\phi(x)\,\mathrm{d}x \text{ for all } \phi \in C_0^\infty(\Omega).$$

The following theorem couples both classical and weak derivative.

Theorem B.4
For $f \in C^m(\Omega)$ and all $|\beta| \leq m \in \mathbb{N}_0$ the classical derivative $D^\beta f$ is also the weak β-th derivative of f.

As a result we can use the term $D^\beta f$ for the weak derivative of f as well. Now, the definition of weak derivative leads us directly to the definition of Sobolev spaces.

Definition B.5 (Sobolev spaces)
For $|\beta| \leq m \in \mathbb{N}_0$, $1 \leq p \leq \infty$ and $\Omega \subset \mathbb{R}^d$ we define the *Sobolev space* $W^{m,p}(\Omega)$ as space of all functions $f \in L^p(\Omega)$, so that the weak β-th derivative $D^\beta f$ exists with $D^\beta f \in L^p(\Omega)$. The corresponding Sobolev norms are given by

$$||f||_{W^{m,p}(\Omega)} := \left(\sum_{\beta \leq m} ||D^\beta f||^p_{L^p(\Omega)} \right)^{\frac{1}{p}} \text{ for } 1 \leq p < \infty,$$
$$||f||_{W^{m,\infty}(\Omega)} := \max_{|\beta| \leq m} ||D^\beta f||_{L^\infty(\Omega)}.$$

C Some special functions and their properties

We introduce some special functions and their properties. For an advanced treatment of special functions we refer to [131, 132] and adopt the notation therein.

Definition C.1
We define for all $n \in \mathbb{N}_0$ and $\tau > 0$

a) the *Pochhammer function*

$$(\tau)_n := \prod_{k=1}^{n} (\tau + k - 1) = \tau \cdot (\tau + 1) \cdot \ldots \cdot (\tau + n - 1),$$
$$(\tau)_0 := 1,$$

b) and the *Gamma function*

$$\Gamma(\tau) := \int_0^\infty \mathrm{e}^{-t} \cdot t^{\tau - 1} \mathrm{d}t.$$

The following theorem shows some basic properties and relation of both functions.

Theorem C.2
For $n, m \in \mathbb{N}_0$ and $\tau > 0$ we have

a) $(\tau)_n = \Gamma(\tau + n) \cdot \Gamma(\tau)^{-1}$,

b) $(\tau)_n \cdot (\tau + n) = (\tau)_{n+1}$,

c) $\Gamma(n) = (n-1)! = \Gamma(n-1) \cdot (n-1)$,

d) $\Gamma(n + m) \geq \Gamma(n) \cdot \Gamma(m + 1)$.

Proof. Proofs of these properties are either simple or can be found in [132]. □

The following definition introduces Gegenbauer polynomials by their generating relation. These polynomials generalize the well-known Legendre polynomials, which are often used in the context of multipole expansion theory [90].

Definition C.3 (Gegenbauer polynomials)
For $n \in \mathbb{N}_0$, $x \in \mathbb{R}$ and $\tau > -\frac{1}{2}$ we define the *Gegenbauer polynomials* $\mathcal{C}_n^a$ by their formal generating relation

$$\frac{1}{(1-2xt+t^2)^\tau} = \sum_{n=0}^{\infty} \mathcal{C}_n^\tau(x) t^n.$$

Their most important properties are listed in the next theorem.

Theorem C.4
Let $n \in \mathbb{N}_0$ and $\tau > 0$.

a) For all $x \in \mathbb{R}$ we have

$$\mathcal{C}_n^\tau(x) = \sum_{k=0}^{\lfloor \frac{j}{2} \rfloor} \frac{(-1)^k \cdot (\tau)_{n-k}}{(2x)^{n-2k} \cdot (n-2k)! \cdot k!}.$$

b) For $|x| \leq 1$ the following inequality holds:

$$|\mathcal{C}_n^\tau(x)| \leq \mathcal{C}_n^\tau(1) = \frac{(2\tau)_n}{n!}.$$

c) For all $x \in \mathbb{R}$ we have

$$\frac{\mathrm{d}}{\mathrm{d}x} \mathcal{C}_n^\tau(x) = 2a \cdot \mathcal{C}_{n-1}^{\tau+1}(x).$$

Proof. Proofs and derivations can be found in [131] and [132]. □

Next, we introduce the hypergeometric series and function, which help us solving and estimating a certain type of integrals as shown in the subsequent theorem.

Definition C.5 (Hypergeometric functions)
For $|z| < 1$ and $a, b, c > 0$ we define

$$H(a, b; c; z) := \sum_{n=0}^{\infty} \frac{(a)_n \cdot (b)_n}{(c)_n} \cdot \frac{z^n}{n!}.$$

Here H (often written as ${}_2F_1$) is called a *hypergeometric function*, while the right-hand side is the *hypergeometric series*.

The following theorem cites two important integral expressions.

Theorem C.6

a) If $|\alpha| < 1$ and $b, c > 0$, then

$$\int_0^1 t^{b-1}(1-t)^{c-1}(1-\alpha t)^{-a}\mathrm{d}t = \frac{\Gamma(b+c)}{\Gamma(b)\Gamma(c)} \cdot H(a, b; b+c; \alpha).$$

b) For $2b > a > 0$ we have

$$\int_0^\infty t^{a-1}(1+t^2)^{-b}\mathrm{d}t = \frac{1}{2}\frac{\Gamma\left(\frac{a}{2}\right)\Gamma\left(b-\frac{a}{2}\right)}{\Gamma(b)}.$$

Proof. These relations are taken from [92]. Proofs can be found in the references therein. □

Bibliography

[1] **H. Helmholtz**, On integrals of the hydrodynamical equations, which express vortex-motion, *Philosophical Magazine*, 33 (226), 485–512 (1867).

[2] **G. K. Batchelor**, *An Introduction to Fluid Dynamics*, Cambridge University Press (1967).

[3] **A. J. Chorin** and **J. E. Marsden**, *A Mathematical Introduction to Fluid Mechanics*, 2nd edition, Springer (1990).

[4] **L. D. Landau** and **E. M. Lifshitz**, *Course of Theoretical Physics, Vol. 6: Fluid Mechanics*, 2nd edition, Butterworth Heinemann (1987).

[5] **R. Panton**, *Incompressible Flow*, 2nd edition, John Wiley & Sons Inc (1996).

[6] **H. Lamb**, *Hydrodynamics*, Cambridge University Press (1895).

[7] **O. A. Ladyzhenskaya**, *Mathematical Theory of Viscous Incompressible Flow*, 2nd edition, Gordon & Breach Science Publishers Ltd (1969).

[8] **R. Temam**, *Navier-Stokes Equations: Theory and Numerical Analysis*, 3rd edition, Elsevier Science Ltd (1985).

[9] **C. Y. Wang**, Exact solutions of the unsteady Navier-Stokes equations, *Applied Mechanics Reviews*, 42 (11), 269–282 (1989).

[10] **C. Y. Wang**, Exact solutions of the steady-state Navier-Stokes equations, *Annual Review of Fluid Mechanics*, 23 (1), 159–177 (1991).

[11] **M. Gellert** and **R. Harbord**, A stable vorticity-velocity formulation for viscous flow analysis, *Computational Mechanics*, 5 (6), 417–427 (1990).

[12] **C. G. Speziale**, On the advantages of the vorticity-velocity formulation of the equations of fluid dynamics, *Journal of Computational Physics*, 73 (2), 476–480 (1987).

[13] **G.-H. Cottet** and **P. Koumoutsakos**, *Vortex Methods: Theory and Applications*, 2nd edition, Cambridge University Press (2000).

[14] **A. J. Majda** and **A. L. Bertozzi**, *Vorticity and Incompressible Flow*, Cambridge University Press (2001).

[15] **L. Ting** and **R. Klein**, *Viscous Vortical Flows*, Springer (1991).

[16] **J.-Z. Wu**, **H.-Y. Ma** and **J. Z. Zhou**, *Vorticity and Vortex Dynamics*, 1st edition, Springer, Berlin (2006).

[17] **G. Winckelmans** and **A. Leonard**, Contributions to vortex particle methods for the computation of three-dimensional incompressible unsteady flows, *Journal of Computational Physics*, 109 (2), 247–273 (1993).

[18] **O. M. Knio** and **A. F. Ghoniem**, Numerical study of a three-dimensional vortex method, *Journal of Computational Physics*, 86 (1), 75–106 (1990).

[19] **G. Barton**, *Elements of Green's Functions and Propagation: Potentials, Diffusion, and Waves*, Oxford University Press (1989).

[20] **J. D. Jackson**, *Classical Electrodynamics*, 3rd edition, John Wiley & Sons (1998).

[21] **A. Leonard**, Vortex methods for flow simulation, *Journal of Computational Physics*, 37 (3), 289–335 (1980).

[22] **A. Leonard**, Computing three-dimensional incompressible flows with vortex elements, *Annual Review of Fluid Mechanics*, 17, 523–559 (1985).

[23] **F. Ponta**, The KLE method: a velocity-vorticity formulation for the Navier-Stokes equations, *Journal of Applied Mechanics, Transactions of the ASME*, 73, 1031–1038 (2006).

[24] **J. D. Anderson**, *Computational Fluid Dynamics*, 6th edition, McGraw-Hill (1995).

[25] **J. H. Ferziger** and **M. Peric**, *Computational Methods for Fluid Dynamics*, 3rd edition, Springer (2010).

[26] **M. D. Gunzburger**, *Finite Element Methods for Viscous Incompressible Flows: A Guide to Theory, Practice, and Algorithms*, Academic Press (1989).

[27] **S. Choudhury** and **R. A. Nicolaides**, Discretization of incompressible vorticity-velocity equations on triangular meshes, *International Journal for Numerical Methods in Fluids*, 11 (6), 823–833 (1990).

[28] **T. T. Medjo**, Vorticity-velocity formulation for the stationary Navier-Stokes equations: the three-dimensional case, *Applied Mathematics Letters*, 8 (4), 63–66 (1995).

[29] **G. Zumbusch**, *Parallel Multilevel Methods: Adaptive Mesh Refinement and Loadbalancing*, 1 edition, Vieweg+Teubner (2003).

[30] **J. J. Monaghan**, An introduction to SPH, *Computer Physics Communications*, 48 (1), 89–96 (1988).

[31] **J. J. Monaghan**, Smoothed particle hydrodynamics, *Annual Review of Astronomy and Astrophysics*, 30, 543–574 (1992).

[32] **G. S. Winckelmans**, *Topics in vortex methods for the computation of three- and two-dimensional incompressible unsteady flows*, Ph.D. thesis, California Institute of Technology, USA (1989).

[33] **G.-H. Cottet**, A new approach for the analysis of vortex methods in two and three dimensions, Technical Report 3, Annales de l'institut Henri Poincaré (C) (1988).

[34] **G. Winckelmans** and **A. Leonard**, Weak solutions of the three-dimensional vorticity equation with vortex singularities, *Physics of Fluids*, 31 (7), 1838–1839 (1988).

[35] **C. Rehbach**, Numerical calculation of three-dimensional unsteady flows with vortex sheets, in *AIAA 16th Aerospace Sciences Meeting*, 289–298, Huntsville, Alabama, USA (1978).

[36] **L. Rosenhead**, The formation of vortices from a surface of discontinuity, *Proceedings of the Royal Society of London. Series A*, 134 (823), 170–192 (1931).

[37] **T. Y. Hou** and **J. Lowengrub**, Convergence of the point vortex method for the 3-D Euler equations, *Communications on Pure and Applied Mathematics*, 43 (8), 965–981 (1990).

[38] **J. Goodman**, **T. Y. Hou** and **J. Lowengrub**, Convergence of the point vortex method for the 2-D Euler equations, *Communications on Pure and Applied Mathematics*, 43 (3), 415–430 (1990).

[39] **J. T. Beale** and **A. Majda**, Vortex methods. I – Convergence in three dimensions. II – Higher order accuracy in two and three dimensions, *Mathematics of Computation*, 39, 1–52 (1982).

[40] **A. J. Chorin**, Numerical study of slightly viscous flow, *Journal of Fluid Mechanics*, 57 (4), 785–796 (1973).

[41] **A. J. Chorin** and **P. S. Bernard**, Discretization of a vortex sheet, with an example of roll-up, *Journal of Computational Physics*, 13 (3), 423–429 (1973).

[42] **O. Hald** and **V. M. D. Prete**, Convergence of vortex methods for Euler's equations, *Mathematics of Computation*, 32 (143), 791–809 (1978).

[43] **J. T. Beale** and **A. Majda**, High order accurate vortex methods with explicit velocity kernels, *Journal of Computational Physics*, 58 (2), 188–208 (1985).

[44] **M. Perlman**, On the accuracy of vortex methods, *Journal of Computational Physics*, 59 (2), 200–223 (1985).

[45] **L. Rosenhead**, The spread of vorticity in the wake behind a cylinder, *Proceedings of the Royal Society of London. Series A*, 127 (806), 590–612 (1930).

[46] **O. H. Hald**, Convergence of vortex methods for Euler's equations, III, *SIAM Journal on Numerical Analysis*, 24 (3), 538–582 (1987).

[47] **H. O. Nordmark**, *Higher order vortex methods with rezoning*, Ph.D. thesis, University of California, Berkeley, USA (1988).

[48] **D. Fishelov**, A new vortex scheme for viscous flows, *Journal of Computational Physics*, 86 (1), 211–224 (1990).

[49] **L. A. Barba**, *Vortex Method for computing high-Reynolds number flows: Increased accuracy with a fully mesh-less formulation.*, Ph.D. thesis, California Institute of Technology, USA (2004).

[50] **K. Atkinson** and **W. Han**, *Theoretical Numerical Analysis: A Functional Analysis Framework*, Springer (2001).

[51] **K. Yosida**, *Functional Analysis*, 6th edition, Springer (1995).

[52] **P. Koumoutsakos**, **G.-H. Cottet** and **D. Rossinelli**, Flow simulations using particles: bridging computer graphics and CFD, in *ACM SIGGRAPH 2008 classes*, 1–73, ACM, Los Angeles, California, USA (2008).

[53] **J. P. Choquin** and **B. Lucquin-Desreux**, Accuracy of a deterministic particle method for Navier-Stokes equations, *International Journal for Numerical Methods in Fluids*, 8 (11), 1439–1458 (1988).

[54] **L. A. Barba**, **A. Leonard** and **C. B. Allen**, Advances in viscous vortex methods – meshless spatial adaption based on radial basis function interpolation, *International Journal for Numerical Methods in Fluids*, 47, 387–421 (2005).

[55] **G. Winckelmans**, **J. K. Salmon**, **M. S. Warren**, **A. Leonard** and **B. Jodoin**, Application of fast parallel and sequential tree codes to computing three-dimensional flows with the vortex element and boundary element methods, *ESAIM: Proceedings*, 1, 225–240 (1996).

[56] **J. H. Williamson**, Low-storage Runge-Kutta schemes, *Journal of Computational Physics*, 35 (1), 48–56 (1980).

[57] **J. T. Beale**, On the accuracy of vortex methods at large times, in *Computational Fluid Dynamics and Reacting Gas Flows*, edited by **B. Engquist et al.**, Springer (1988).

[58] **H. O. Nordmark**, Rezoning for higher order vortex methods, *Journal of Computational Physics*, 97 (2), 366–397 (1991).

[59] **P. Koumoutsakos**, Inviscid axisymmetrization of an elliptical vortex, *Journal of Computational Physics*, 138 (2), 821–857 (1997).

[60] **I. J. Schoenberg**, *Cardinal spline interpolation*, Society for Industrial and Applied Mathematics (1973).

[61] **R. W. Hockney** and **J. W. Eastwood**, *Computer Simulation Using Particles*, Taylor & Francis (1989).

[62] **S. E. Hieber** and **P. Koumoutsakos**, A Lagrangian particle level set method, *Journal of Computational Physics*, 210 (1), 342–367 (2005).

[63] **C. K. Birdsall** and **D. Fuss**, Clouds-in-clouds, clouds-in-cells physics for many-body plasma simulation, *Journal of Computational Physics*, 3 (4), 494–511 (1969).

[64] **I. P. Christiansen**, Numerical simulation of hydrodynamics by the method of point vortices, *Journal of Computational Physics*, 13 (3), 363–379 (1973).

[65] **G.-H. Cottet**, **M. L. O. Salihi** and **M. E. Hamraoui**, Multi-purpose regridding in vortex methods, *ESAIM: Proceedings*, 7 (1998).

[66] **J. J. Monaghan**, Extrapolating B-splines for interpolation, *Journal of Computational Physics*, 60 (2), 253–262 (1985).

[67] **A. K. Chaniotis**, **D. Poulikakos** and **P. Koumoutsakos**, Remeshed smoothed particle hydrodynamics for the simulation of viscous and heat conducting flows, *Journal of Computational Physics*, 182 (1), 67–90 (2002).

[68] **M. Coquerelle** and **G.-H. Cottet**, A vortex level set method for the two-way coupling of an incompressible fluid with colliding rigid bodies, *Journal of Computational Physics*, 227 (21), 9121–9137 (2008).

[69] **P. Ploumhans**, **G. Winckelmans**, **J. K. Salmon**, **A. Leonard** and **M. S. Warren**, Vortex methods for direct numerical simulation of three-dimensional bluff body flows: Application to the sphere at Re=300, 500, and 1000, *Journal of Computational Physics*, 178 (2), 427–463 (2002).

[70] **R. Cocle**, **G. Winckelmans** and **G. Daeninck**, Combining the vortex-in-cell and parallel fast multipole methods for efficient domain decomposition simulations, *Journal of Computational Physics*, 227 (21), 9091–9120 (2008).

[71] **P. Degond** and **S. Mas-Gallic**, The weighted particle method for convection-diffusion equations, Part 1: The case of an isotropic viscosity, *Mathematics of Computation*, 53 (188), 485–507 (1989).

[72] **S. Shankar** and **L. L. van Dommelen**, A new diffusion scheme in vortex methods for three-dimensional incompressible flows, *ESAIM: Proceedings*, 1, 587–599 (1996).

[73] **P. Ploumhans** and **G. S. Winckelmans**, Vortex methods for high-resolution simulations of viscous flow past bluff bodies of general geometry, *Journal of Computational Physics*, 165 (2), 354–406 (2000).

[74] **S. Shankar**, *A new mesh-free vortex method*, Ph.D. thesis, Florida State University, USA (1996).

[75] **S. Shankar** and **L. L. van Dommelen**, A new diffusion procedure for vortex methods, *Journal of Computational Physics*, 127 (1), 88–109 (1996).

[76] **D. Wee** and **A. F. Ghoniem**, Modified interpolation kernels for treating diffusion and remeshing in vortex methods, *Journal of Computational Physics*, 213 (1), 239–263 (2006).

[77] **P. G. Saffman**, *Vortex Dynamics*, Cambridge University Press (1995).

[78] **H. Fangohr**, Efficient methods for handling long-range forces in particle-particle simulations, *Journal of Computational Physics*, 162 (2), 372–384 (2000).

[79] **P. Steinbach** and **B. Brooks**, New spherical-cutoff methods for long-range forces in macromolecular simulation, *Journal of Computational Chemistry*, 15 (7), 667–683 (1994).

[80] **D. Wolf**, **P. Keblinski**, **S. R. Phillpot** and **J. Eggebrecht**, Exact method for the simulation of Coulombic systems by spherically truncated, pairwise r^{-1} summation, *The Journal of Chemical Physics*, 110 (17), 8254–8282 (1999).

[81] **P. Brommer**, **P. Beck**, **A. Chatzopoulos**, **F. Gähler**, **J. Roth** and **H. Trebin**, Direct Wolf summation of a polarizable force field for silica, *The Journal of Chemical Physics*, 132 (19), 194109–1 (2010).

[82] **J. E. Barnes** and **P. Hut**, A hierarchical $\mathcal{O}(N \log N)$ force-calculation algorithm, *Nature*, 324 (6096), 446–449 (1986).

[83] **L. Greengard** and **V. Rokhlin**, A fast algorithm for particle simulations, *Journal of Computational Physics*, 73 (2), 325–348 (1987).

[84] **F. N. Beg**, **M. S. Wei**, **E. L. Clark**, **A. E. Dangor**, **R. G. Evans**, **P. Gibbon**, **A. Gopal**, **K. L. Lancaster**, **K. W. D. Ledingham**, **P. McKenna**, **P. A. Norreys**, **M. Tatarakis**, **M. Zepf** and **K. Krushelnick**, Return current and proton emission from short pulse laser interactions with wire targets, in *45th Annual Meeting of the APS Division of Plasma Physics*, volume 11, 2806–2813, AIP, Albuquerque, New Mexico, USA (2004).

[85] **N. A. Gumerov** and **R. Duraiswami**, *Fast Multipole Methods for the Helmholtz Equation in Three Dimensions*, Elsevier Science (2005).

[86] **P. Miocchi** and **R. Capuzzo-Dolcetta**, An efficient parallel tree-code for the simulation of self-gravitating systems, *Astronomy and Astrophysics*, 382 (2), 758–767 (2002).

[87] **S. Pfalzner** and **P. Gibbon**, A 3D hierarchical tree code for dense plasma simulation, *Computer Physics Communications*, 79 (1), 24–38 (1994).

[88] **J. Wambsganss**, Gravitational lensing: numerical simulations with a hierarchical tree code, *Journal of Computational and Applied Mathematics*, 109 (1–2), 353–372 (1999).

[89] **S. Pfalzner** and **P. Gibbon**, *Many-Body Tree Methods in Physics*, Cambridge University Press (1996).

[90] **J. K. Salmon**, *Parallel hierarchical N-body methods*, Ph.D. thesis, California Institute of Technology, USA (1991).

[91] **C. A. White** and **M. Head-Gordon**, Derivation and efficient implementation of the fast multipole method, *Journal of Chemical Physics*, 101, 6593–6605 (1994).

[92] **I. S. Gradshteyn** and **I. M. Ryzhik**, *Table of Integrals, Series and Products*, 6th edition, Academic Press (2000).

[93] **S. Chapman**, On the expansion of $(1 - 2r\cos\theta + r^2)^n$ in a series of Legendre's functions, *The Quarterly Journal of Pure and Applied Mathematics*, 185, 16–26 (1916).

[94] **J. E. Barnes** and **P. Hut**, Error analysis of a tree code, *Astrophysical Journal Supplement Series*, 70, 389–417 (1989).

[95] **J. K. Salmon** and **M. S. Warren**, Skeletons from the treecode closet, *Journal of Computational Physics*, 111 (1), 136–155 (1994).

[96] **I. Chowdhury** and **V. Jandhyala**, Single level multipole expansions and operators for potentials of the form $r^{-\lambda}$, *SIAM Journal on Scientific Computing*, 26 (3), 930–943 (2005).

[97] **Z.-H. Duan** and **R. Krasny**, An adaptive treecode for computing nonbonded potential energy in classical molecular systems, *Journal of Computational Chemistry*, 22 (2), 184–195 (2001).

[98] **K. Srinivasan**, **H. Mahawar** and **V. Sarin**, A multipole based treecode using spherical harmonics for potentials of the form $r^{-\lambda}$, in *5th International Conference of Computational Science – ICCS 2005*, 107–114, Atlanta, Georgia, USA (2005).

[99] **J. K. Salmon**, **M. S. Warren** and **G. Winckelmans**, Fast parallel tree codes for gravitational and fluid dynamical N-body problems, *The International Journal of Supercomputer Applications*, 8, 129–142 (1994).

[100] **D. Wee**, **Y. M. Marzouk**, **F. Schlegel** and **A. F. Ghoniem**, Convergence characteristics and computational cost of two algebraic kernels in vortex methods with a tree-code algorithm, *SIAM Journal on Scientific Computing*, 31 (4), 2510–2527 (2009).

[101] **K. Lindsay** and **R. Krasny**, A particle method and adaptive treecode for vortex sheet motion in three-dimensional flow, *Journal of Computational Physics*, 172 (2), 879–907 (2001).

[102] **P. Gibbon**, PEPC Homepage, http://www.fz-juelich.de/jsc/pepc/ (2010).

[103] **L. Hernquist**, Hierarchical N-body methods, *Computer Physics Communications*, 48 (1), 107–115 (1988).

[104] **H. Dachsel**, Corrected article: "An error-controlled fast multipole method" [J. Chem. Phys. 131, 244102 (2009)], *The Journal of Chemical Physics*, 132 (11), 119901 (2010).

[105] **I. Kabadshow**, *Periodic Boundary Conditions and the Error-Controlled Fast Multipole Method*, Ph.D. thesis, Bergische Universität Wuppertal, Germany (2010).

[106] **P. Gibbon**, PEPC: Pretty efficient parallel coulomb solver, Technical Report FZJ-ZAM-IB-2003-05, Jülich Supercomputing Centre, Germany (2003).

[107] **R. Speck**, **L. Arnold** and **P. Gibbon**, Towards a petascale tree code: Scaling and efficiency of the PEPC library, *Journal of Computational Science* (2010, submitted).

[108] **R. Speck**, **P. Gibbon** and **M. Hoffmann**, Efficiency and scalability of the parallel Barnes-Hut tree code PEPC, in *Parallel Computing: From Multicores and GPU's to Petascale*, edited by **B. Chapman**, **F. Desprez**, **G. R. Joubert**, **A. Lichnewsky**, **F. Peters** and **T. Priol**, volume 19, Lyon, France (2010).

[109] **P. Gibbon**, **R. Speck**, **A. Karmakar**, **L. Arnold**, **W. Frings**, **B. Berberich**, **D. Reiter** and **M. Masek**, Progress in mesh-free plasma simulation with parallel tree codes, *IEEE Transactions on Plasma Science*, 38 (9), 2367–2376 (2010).

[110] **M. S. Warren** and **J. K. Salmon**, A parallel hashed oct-tree N-body algorithm, in *Proceedings of the 1993 ACM/IEEE conference on Supercomputing*, 12–21, Portland, Oregon, USA (1993).

[111] **M. S. Warren** and **J. K. Salmon**, A portable parallel particle program, *Computer Physics Communications*, 87, 266–290 (1995).

[112] **H. Sagan**, *Space-Filling Curves*, 1st edition, Springer (1994).

[113] **P. Gibbon**, **M. Hofmann**, **G. Ruenger** and **R. Speck**, Parallel sorting algorithms for optimizing particle simulations, in *2010 IEEE International Conference on Cluster Computing, Workshops and Posters (Cluster Workshops)*, 1–8, IEEE, Heraklion, Greece (2010).

[114] **J. Bruck**, **C. Ho**, **S. Kipnis**, **E. Upfal** and **D. Weathersby**, Efficient algorithms for All-to-All communications in multi-port message-passing systems, *IEEE Transactions on parallel and distributed systems*, 8, 298–309 (1997).

[115] **M. Hofmann**, private communication (2010).

[116] **E. B. Saff** and **A. B. J. Kuijlaars**, Distributing many points on a sphere, *The Mathematical Intelligencer*, 19 (1), 5–11 (1997).

[117] **F. A. Cruz**, **M. G. Knepley** and **L. A. Barba**, PetFMM – A dynamically load-balancing parallel fast multipole library, *International Journal for Numerical Methods in Engineering* (2010).

[118] **M. Frigo** and **S. G. Johnson**, FFTW: an adaptive software architecture for the FFT, in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 3, 1381–1384 (1998).

[119] **P. Chatelain**, **A. Curioni**, **M. Bergdorf**, **D. Rossinelli**, **W. Andreoni** and **P. Koumoutsakos**, Billion vortex particle direct numerical simulations of aircraft wakes, *Computer Methods in Applied Mechanics and Engineering*, 197 (13–16), 1296–1304 (2008).

[120] **P. R. Schatzle**, *An experimental study of fusion of vortex rings*, Ph.D. thesis, California Institute of Technology, USA (1987).

[121] **I. Naoki**, **O. Koichi** and **O. Yuko**, Experimental study of interacting vortex rings, *The Institute of Space and Astronautical Science Report*, 5, 3–13 (1987).

[122] **S. Kida**, **M. Takaoka** and **F. Hussain**, Collision of two vortex rings, *Journal of Fluid Mechanics*, 230, 583–646 (1991).

[123] **P. Koumoutsakos**, Multiscale flow simulations using particles, *Annual Review of Fluid Mechanics*, 37 (1), 457–487 (2005).

[124] **J. D. Eldredge**, **A. Leonard** and **T. Colonius**, A general deterministic treatment of derivatives in particle methods, *Journal of Computational Physics*, 180 (2), 686–709 (2002).

[125] **P. P. Ewald**, Die Berechnung optischer und elektrostatischer Gitterpotentiale, *Annalen der Physik*, 369 (3), 253–287 (1921).

[126] **C. Pozrikidis**, Theoretical and computational aspects of the self-induced motion of three-dimensional vortex sheets, *Journal of Fluid Mechanics*, 425, 335–366 (2000).

[127] **M. Challacombe**, **C. White** and **M. Head-Gordon**, Periodic boundary conditions and the fast multipole method, *The Journal of Chemical Physics*, 107, 10131–10140 (1997).

[128] **K. N. Kudin** and **G. E. Scuseria**, Revisiting infinite lattice sums with the periodic fast multipole method, *The Journal of Chemical Physics*, 121, 2886–2890 (2004).

[129] **L. Koenigsberger**, *Hermann von Helmholtz*, Clarendon Press, Oxford (1906).

[130] **G. B. Arfken** and **H. Weber**, *Mathematical Methods for Physicists*, 5th edition, Academic Press Inc (2001).

[131] **A. Erdelyi** (Editor), *Higher Transcendental Functions*, volume 1–3 of *Bateman Manuscript Project*, McGraw-Hill Book Company (1953).

[132] **E. D. Rainville**, *Special functions*, Macmillan (1960).